บทที่ 3

การออกแบบตัวควบคุมลำดับที่โปรแกรมได้ (PC)

แนวความคิดในการออกแบบตัวควบคุมแบบลำดับที่โปรแกรมได้ สามารถแบ่งการออกแบบออกเป็น 3 ส่วนใหญ่ ๆ คือ การออกแบบระบบ การออกแบบทางฮาร์ดแวร์และการออกแบบทางซอฟท์แวร์ ในบทนี้จะกล่าวถึงแนวความคิด และหลักการอย่างกว้าง ๆ ในการออกแบบ ในส่วนของรายละเอียดจะกล่าวถึงในบทต่อไป

3.1 แนวความคิดในการออกแบบระบบ

การออกแบบระบบของ PC จะต้องกำหนดสเปค เช่น จำนวนอินพุท เอาท์พุท (I/O) ของตัวควบคุม (Control Unit) จำนวนอินพุทและเอาท์พุทที่สามารถจะขยายได้สูงสุด ขนาดความจุของชั้นโปรแกรมควบคุม เพื่อที่จะหาขนาดของตัว ซีพียู ที่จะนำมาใช้ควบคุมได้ถูกต้อง จากการศึกษา PC ที่มีอยู่ในท้องตลาดพบว่าขนาดของ PC ที่มีผู้นิยมใช้อย่างแพร่หลาย จะมี I/O อยู่ระหว่าง 28-40 จุด ดังนั้นการออกแบบสร้าง PC ในวิทยานิพนธ์นี้จึงกำหนดให้มีจำนวน I/O ในตัวควบคุม 32 จุดและสามารถขยายได้สูงสุดเป็น 64 จุด ซึ่งเพียงพอกับการใช้งานควบคุมโดยทั่วไป การกำหนดให้มีจำนวน I/O น้อยกว่านี้ จะทำให้ระบบมีขนาดเล็กเกินไป จำนวน I/O ไม่เพียงพอต่อการนำไปใช้ในงานควบคุมและจะไม่คุ้มต่อการลงทุนเพราะการสร้าง PC ขึ้นใช้งานจะมีการลงทุนที่คงที่อยู่ส่วนหนึ่ง และราคาจะเพิ่มตามจำนวน I/O ที่เพิ่มขึ้น ดังนั้นในงานควบคุมที่ใช้จำนวน I/O ไม่มากนักควรใช้ระบบรีเลย์เดิมเพราะมีราคาถูกกว่า ส่วนการกำหนดให้ PC มีจำนวน I/O มากกว่า 64 จุดจะทำให้ตัวควบคุมมีขนาดใหญ่ จะต้องสร้างในลักษณะที่เป็นโมดูลซึ่งเหมาะสำหรับ PC ขนาดกลางและขนาดใหญ่ ทำให้ผิดวัตถุประสงค์ของการทำวิทยานิพนธ์นี้ที่ต้องการ PC ขนาดเล็ก นอกจากนี้จะทำให้ต้องเพิ่มจำนวนชั้นโปรแกรมเพื่อให้เป็นสัดส่วนกับจำนวน I/O ที่เพิ่มขึ้น การเพิ่มจำนวนชั้นโปรแกรมให้มากขึ้นอาจทำให้ความเร็วในการทำโปรแกรมชั้นบันไดไม่เพียงพอ ทำให้ต้องเพิ่มขนาดของตัว ซีพียู ทำให้ขนาด

ของระบบใหญ่และมีความซับซ้อนมากขึ้น

นอกจากนี้ยังจะต้องคำนึงถึงความเร็วของระบบ โดยเวลาของซีพียูส่วนใหญ่จะใช้ในการประมวลผลโปรแกรมของผู้ใช้และจะต้องใช้เวลาส่วนน้อยในการบริการอุปกรณ์Peripheral ดังนั้นอุปกรณ์ Peripheral ที่ซีพียูจะต้องใช้เวลาในการบริการนาน เช่น ตัวป้อนโปรแกรม ซีพียูต้องเสียเวลามาสแกนคีย์บอร์ดและตัวแสดงผล ดังนั้นในการออกแบบตัวป้อนโปรแกรมจึงใช้ฮาร์ดแวร์มาช่วยในการสแกนคีย์บอร์ดและตัวแสดงผลแทนซีพียู และจุดที่จะต้องคำนึงถึงในการออกแบบอีกจุดหนึ่งคือสถานที่ที่จะนำเครื่องไปใช้งาน ตัว PC ที่ออกแบบมีวัตถุประสงค์ในการนำไปใช้ควบคุมเครื่องจักรในโรงงานอุตสาหกรรมซึ่งสภาพแวดล้อมเต็มไปด้วยสัญญาณรบกวน การออกแบบจึงต้องมีการป้องกันสัญญาณรบกวนเหล่านี้ด้วยโดย

1. ป้องกันซีพียูทำงานออกนอกโปรแกรมโดยใช้วอทช์ดอกไทเมอร์เป็นตัวรีเซทให้ซีพียูกลับมาเริ่มทำงานใหม่

2. ป้องกันระบบทำงานผิดพลาดเนื่องจากไฟตก โดยใช้พาวเวอร์มอนิเตอร์เป็นตัวเฝ้าระดับแรงดันไฟ เมื่อระดับแรงดันไฟตกลงต่ำกว่าที่อุปกรณ์ IC จะทำงานได้ถูกต้อง ตัวพาวเวอร์มอนิเตอร์จะรีเซทซีพียูไม่ให้ทำงานจนกว่าระดับแรงดันไฟจะถูกต้อง

3. ป้องกันการเหนี่ยวนำของสัญญาณรบกวนจากภายนอก โดยการออกแบบลายแผ่นวงจรพิมพ์ให้มี Ground Plane ล้อมรอบอุปกรณ์ IC มากที่สุดเท่าที่จะทำได้

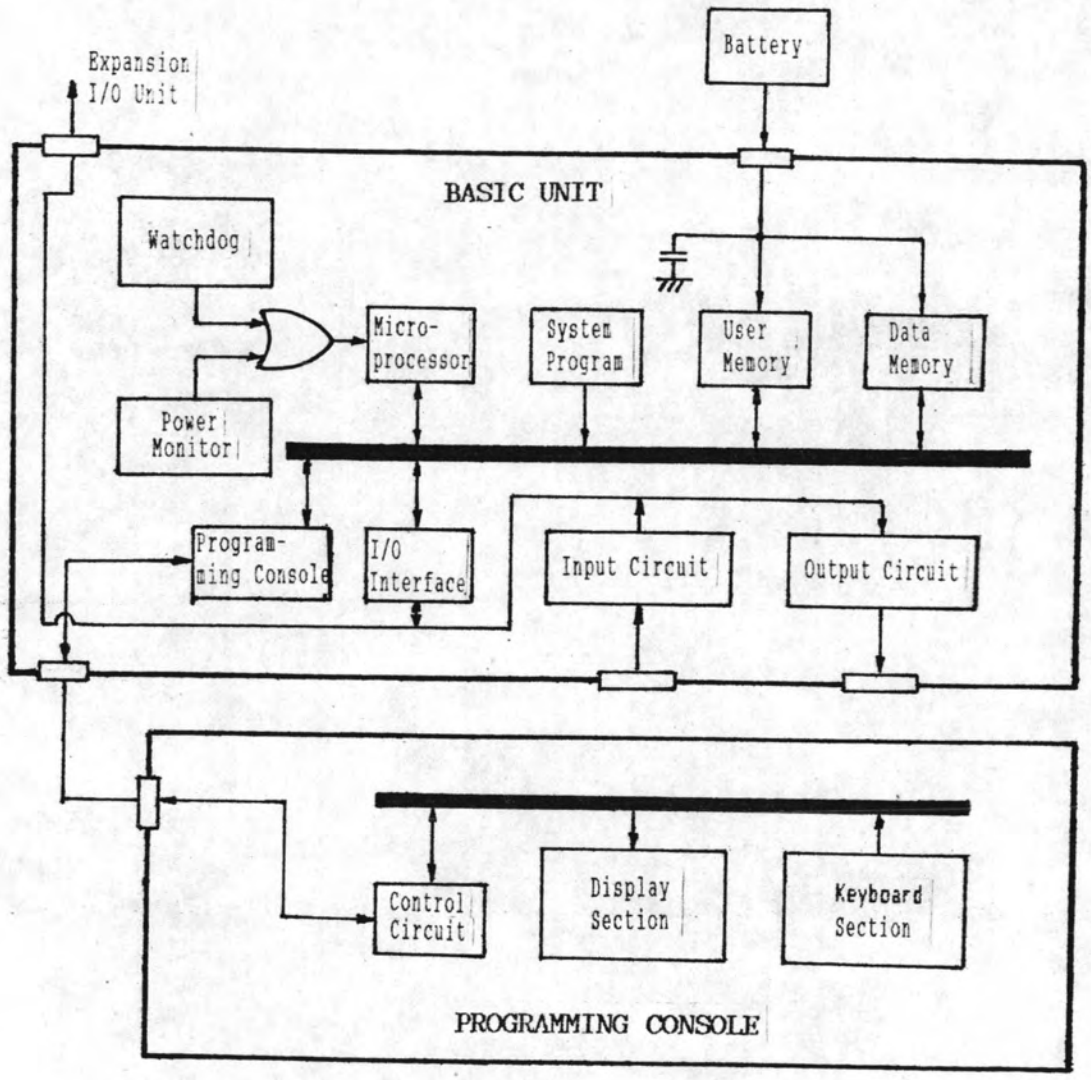
4. ป้องกันการเหนี่ยวนำของสัญญาณรบกวนจากแหล่งจ่ายไฟ โดยใช้ Line Filter ช่วย

สรุปสเปคของ PC

- ระบบการควบคุม	ใช้ไมโครโปรเซสเซอร์ขนาด 8 บิท
- จำนวนอินพุท	16 จุด ชนิด DC กระแสไหลเข้า
- จำนวนเอาท์พุท	16 จุด ชนิดรีเลย์
- จำนวน I/O ที่ขยาย	32 จุด
- ตัวป้อนโปรแกรม	ใช้ LED และ LED 7 ขีดแสดงผลมีคีย์บอร์ดใช้ป้อนโปรแกรม
- ระบบการโปรแกรม	ใช้โปรแกรมภาษาขั้นบันได (Ladder Diagram)
- ความยาวของคำสั่ง	4 Byte/Step
- คำสั่งพื้นฐาน	มีคำสั่ง LD, LD-NOT, AND, AND-NOT, OR, OR-NOT, OUT, AND-BLOCK, OR-BLOCK END, TIM(Timer), CNT(Counter), DS(Data Set), SFT(Shift), PLS(Pulse)

3.2 แนวความคิดในการออกแบบฮาร์ดแวร์

ในบทที่ 2 ได้กล่าวถึงปัญหาที่เกิดขึ้นในการออกแบบ PC ปัญหาใหญ่ที่พบคือ ปัญหาทางด้านความเร็วในการประมวลผลโปรแกรมของผู้ใช้หรือโปรแกรมขั้นบันได (Ladder Diagram) ของซีพียู (CPU) เนื่องจากซีพียูต้องไปจัดการกับโปรแกรมในส่วนอื่น ๆ อีกด้วย เช่น ซีพียูจะต้องมาจัดการโปรแกรมในส่วนแสดงผล และรับคำสั่งจากคีย์บอร์ด ซึ่งเป็นการเพิ่มภาระให้กับซีพียู ดังนั้นเพื่อแก้ปัญหานี้จึงได้นำฮาร์ดแวร์เข้ามาช่วยลดภาระบางส่วนให้แก่ซีพียู การออกแบบแบ่งออกเป็น 2 ส่วนคือส่วนแรกหน่วยควบคุม (Control Unit) หรือที่เรียกว่า BASIC UNIT จะทำหน้าที่ในการอ่านอินพุท แล้วนำมาประมวลผลตามโปรแกรมการควบคุมของโปรแกรมผู้ใช้ ผลลัพธ์ที่ได้จะถูกส่งไปยังเอาท์พุทเพื่อควบคุมระบบที่ต้องการอีกทีหนึ่ง ส่วนที่สองคือ ตัวป้อนโปรแกรม (Programming Console) ทำหน้าที่รับโปรแกรมการควบคุม ตรวจสอบ และแก้ไขโปรแกรม นอกจากนี้ยังใช้ในการตรวจสอบ (Monitor) ดูการทำงานภายในของรีเลย์ ตัวตั้งเวลาและตัวนับโดยใช้ฮาร์ดแวร์ (Control Circuit) เข้ามาช่วยในการสแกนคีย์บอร์ด 8และส่วนแสดงผล ทำให้ ซีพียู ใช้เวลาในการบริการโปรแกรมส่วนนี้สั้นมาก ตัวควบคุมแบบลำดับที่ได้ออกแบบมีส่วนประกอบสำคัญ (ดูรูปที่ 3.1) ดังนี้



รูปที่ 3.1 แสดงบล็อคไดอะแกรมของตัวควบคุมแบบลำดับ

3.2.1 <u>หน่วยควบคุม (Control Unit)</u> ประกอบด้วยบล็อคต่าง ๆ ดังนี้

1. **ไมโครโปรเซสเซอร์ (Microprocessor)** หรือซีพียูเป็นส่วนประมวลผลกลางของระบบ ทำหน้าที่อ่านข้อมูลจากอินพุท แล้วนำมาประมวลผลตามโปรแกรมขั้นบันได (Ladder Diagram) ผลลัพธ์ที่ได้จะถูกส่งไปยังเอาท์พุท เพื่อควบคุมอุปกรณ์กำลังภายนอกอีกทีหนึ่ง ในการออกแบบเลือกใช้หน่วยประมวลผลเบอร์ 8085 ซึ่งเป็นไมโครโปรเซสเซอร์ชนิด 8 บิท มีความถี่ในการทำงาน 6 เมกกะเฮิร์ต (MHz) เหตุผลในการเลือกเนื่องจาก 8085 เป็นไมโครโปรเซสเซอร์ที่ออกแบบมาเพื่อใช้ในการควบคุมโดยเฉพาะ มีระบบบัสเป็นมัลติเพล็กและมีอุปกรณ์สนับสนุนที่ดีสามารถทำให้ระบบที่ออกแบบมีขนาดเล็กลงได้

2. **หน่วยความจำระบบ (System Program)** เป็นหน่วยความจำที่ใช้เก็บโปรแกรมควบคุมการทำงานของเครื่องควบคุมแบบลำดับ (PC) โปรแกรมส่วนนี้จะไม่เปลี่ยนแปลงขณะนำไปใช้งานจริงใช้ EPROM

3. **หน่วยความจำผู้ใช้ (User Memory)** เป็นหน่วยความจำที่ใช้เก็บโปรแกรมขั้นบันได (Ladder Diagram) ที่ผู้ใช้ป้อนเข้าไปโดยผ่านตัวป้อนโปรแกรม (Programming Console) เพื่อส่งไปให้ซีพียูซึ่งจะทำหน้าที่ประมวลผลและส่งสัญญาณไปควบคุมระบบให้ทำงานตามลำดับขั้น (Sequence) ที่ต้องการ โปรแกรมเหล่านี้อาจเปลี่ยนแปลงได้ขณะใช้งานจริงใช้เป็น RAM ความสามารถในการเก็บโปรแกรมขั้นบันได (Ladder Diagram) คิดเป็นขั้นโปรแกรม (Step) ได้ 1000 ขั้น

4. **หน่วยความจำข้อมูล (Data Memory)** เป็นหน่วยความจำที่ใช้เก็บค่าที่ประมวลผลแล้ว เช่น ค่าเวลาของตัวตั้งเวลา ค่านับของตัวนับ และค่าสภาวะการทำงานของรีเลย์ภายใน เป็นต้น ค่าเหล่านี้เปลี่ยนแปลงได้จริงใช้เป็น RAM

5. **วอท์ชดอก ไทเมอร์ (Watchdog Timer)** เป็นวงจรที่ใช้ทำหน้าที่ตรวจสอบการทำงานของซีพียูว่าปกติหรือไม่ถ้าซีพียูทำงานออกนอกโปรแกรมของระบบ เนื่องจากสัญญาณรบกวนหรือสาเหตุอื่นใดก็ตาม วอท์ชดอก ไทเมอร์จะทำการรีเซทให้ซีพียูกลับเข้าสู่ระบบและเริ่มต้นการทำงานใหม่อีกครั้งหนึ่ง วงจรการทำงานของวอท์ชดอก ไทเมอร์ อาจทำได้หลายวิธี เช่น ใช้วงจรโมโนสเตเบิล มัลติไวเบรเตอร์ (Monostable Multivibrator) หรือใช้วงจรอะสเตเบิล มัลติไวเบรเตอร์ (Astable Multivibrator) หลักการทำงานของวอท์ชดอกไทเมอร์คือซีพียูจะต้องส่งสัญญาณมารีเซทวอท์ชดอกไทเมอร์อย่างสม่ำเสมอ เพื่อเป็นการบอก

ให้วอท์ชดอกไทเมอร์รู้ว่าซีพียูยังทำงานตามปกติอยู่ ถ้าไม่มีสัญญาณจากซีพียูมารีเซทไม่ว่าด้วยสาเหตุใดก็ตาม วอท์ชดอกไทเมอร์จะส่งสัญญาณไปรีเซทซีพียูให้เริ่มต้นทำงานใหม่

6. **พาวเวอร์ มอนิเตอร์ (Power Monitor)** เป็นวงจรที่ทำหน้าที่ใช้ตรวจสอบระดับแรงดันที่จ่ายให้กับระบบ วงจรที่ใช้จะเป็น IC ตระกูล TTL และ MOS ซึ่งใช้ไฟเลี้ยงวงจร 5 โวลท์ (Volts) ในกรณีที่ระดับไฟเลี้ยงตกลงต่ำกว่าระดับหนึ่ง ซึ่งจะทำให้ IC ในระบบไม่สามารถทำงานได้ถูกต้อง วงจรพาวเวอร์มอนิเตอร์ (Power Monitor) จะส่งสัญญาณมารีเซทซีพียูไม่ให้ทำงาน จนกว่าระดับไฟเลี้ยงวงจรจะถูกต้อง ซีพียูจึงจะเริ่มทำงานใหม่ นอกจากนี้ยังใช้ร่วมกับหน่วยความจำชนิด RAM โดยทำงานร่วมกับแบตเตอรีเพื่อจ่ายไฟสำรอง (Back up) ให้กับข้อมูลที่อยู่ในหน่วยความจำไม่ให้หายไป

7. **แบตเตอรี (Battery)** เป็นตัวจ่ายไฟเลี้ยงให้กับหน่วยความจำชนิด RAM เพื่อรักษาโปรแกรมของผู้ใช้เครื่องและข้อมูลบางส่วนไม่ให้หายไป เมื่อเวลาไม่ได้จ่ายไฟเลี้ยงให้กับเครื่องและเมื่อไฟตกหรือดับ แบตเตอรีที่ใช้ปกติจะมี 2 ชนิดคือ ชนิดอัดประจุได้ (Rechargeable) เช่น NiCd และชนิดอัดประจุไม่ได้ (Non-Chargeable) เช่น Lithium ในการออกแบบวงจรเลือกใช้ Lithium ซึ่งมีอายุการใช้งานนานถึง 2-3 ปี

8. **วงจรอินพุท (Input Circuit)** [5] เป็นส่วนรับสัญญาณจากสวิทช์และตัวเซนเซอร์ (Sensor) ชนิดต่าง ๆ โดยจะแปลงระดับสัญญาณที่ได้รับมาให้เหมาะสมเพื่อป้อนให้แก่ซีพียูวงจรอินพุทจะต้องมีคุณสมบัติดังนี้

1. ลดสัญญาณรบกวนให้น้อยที่สุด โดยใช้วงจรกรอง เป็นตัวกรองสัญญาณรบกวนออก

2. ป้องกันระบบจากระดับแรงดันเกินจากภายนอก โดยใช้วิธีแยกระบบออกจากกันและติดต่อกันด้วยแสง (Optical Isolation)

3. นอกจากนี้ยังอาจให้มีคุณสมบัติเป็นฮีสเตอร์รีซีส (Hysteresis) เพื่อป้องกันสัญญาณรบกวนที่อาจปะปนผ่านเข้ามาได้

9. **วงจรเอาท์พุท (Output Circuit)** [5] ทำหน้าที่แปลงสัญญาณที่รับมาจากซีพียูให้มีขนาดเหมาะสม เพื่อนำไปขับอุปกรณ์ภายนอก วงจรเอาท์พุทต้องมีคุณสมบัติดังนี้

1. ต้องสามารถป้องกันอัตราการเปลี่ยนแปลงของแรงดัน (dv/dt) ซึ่งเกิดจากโหลดภายนอก โดยใช้วงจรกรองผ่านต่ำ (Low Pass Filter) RC เป็นตัวป้องกัน

2. ต้องสามารถป้องกันแรงดันเสอร์จ (Surge Voltage)ที่เกิดได้โดยใช้ Metal Oxide Varistor

3. ต้องสามารถแยกระบบออกจากเอาท์พุท เพื่อป้องกันระดับแรงดันไฟเกินจากภายนอกโดยใช้ Opto-Isolator

10. **I/O อินเตอร์เฟส (I/O Interface)** เป็นส่วนเชื่อมต่อระหว่าง ซีพียู และ I/O Circuit

11. **ตัวอินเตอร์เฟสตัวป้อนโปรแกรม (Programming Console Interface)** เป็นส่วนเชื่อมต่อระหว่าง ซีพียู และตัวป้อนโปรแกรม

3.2.2 ตัวป้อนโปรแกรม (Programming Console)

ประกอบด้วยบล็อคต่าง ๆ ดังนี้

1. **ส่วนคีย์บอร์ด (Keyboard Section)** เป็นส่วนที่ใช้ในการรับคำสั่งจากผู้ใช้ คีย์บอร์ดที่ใช้มีจำนวน 35 ตัวจะต่อเป็น Matrix ขนาด 5x7 แบ่งเป็น คีย์ตัวเลข (Numeric Key) 10 ปุ่ม คีย์คำสั่ง (Instruction Key) 13 ปุ่ม ที่เหลือใช้เป็น คีย์ควบคุม (Control Key) และอื่น ๆ

2. **ส่วนแสดงผล (Display Section)** เป็นหน่วยแสดงผลใช้ LED แบบ7ขีด9ตัว สามตัวแรกใช้แสดงขั้นโปรแกรม (Step) ซึ่งสามารถแสดงได้ 1000 ขั้น สี่ตัวถัดมาใช้แสดงหมายเลขของรีเลย์ หรือ OPERAND ของคำสั่ง และอีกสองตัวที่เหลือใช้แสดงหมายเลขของฟังค์ชัน นอกจากนี้ยังใช้ LED เป็นตัวแสดงคำสั่งและแสดงโหมดของการทำงาน เช่น READ INSERT DELETE เป็นต้น

3. **วงจรควบคุม (Control Circuit)** ทำหน้าที่ควบคุมการทำงานของภาคคีย์บอร์ด และหน่วยแสดงผล ในการควบคุมนี้ได้ใช้ฮาร์ดแวร์ช่วยในการสแกนคีย์บอร์ด

และหน่วยแสดงผลแทนการสแกนด้วยซอฟท์แวร์ซึ่งทำให้ลดภาระของซีพียูลงได้

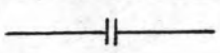
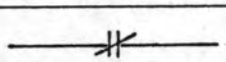
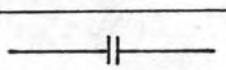
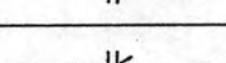
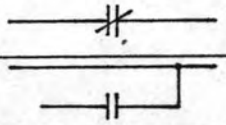
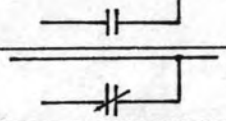
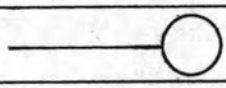
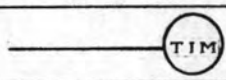
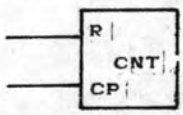
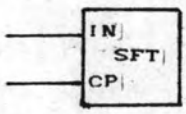
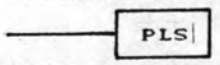
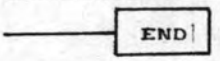
3.3 แนวการออกแบบซอฟท์แวร์

ในบทนี้จะกล่าวถึงแนวความคิดในการออกแบบซอฟท์แวร์ เฉพาะในส่วนวิธีการประมวลผลโปรแกรมขั้นบันได (Ladder Diagram) ของผู้ใช้ ให้มีความสามารถในการประมวลผลเร็วที่สุด โดยการประมวลผลโปรแกรมขั้นบันไดหรือโปรแกรมผู้ใช้นี้ไม่ควรใช้เวลาเกิน 100 msec. ในแต่ละรอบของการทำงาน ในส่วนรายละเอียดของโปรแกรมอื่น ๆ จะกล่าวถึงในบทต่อ ๆ ไป

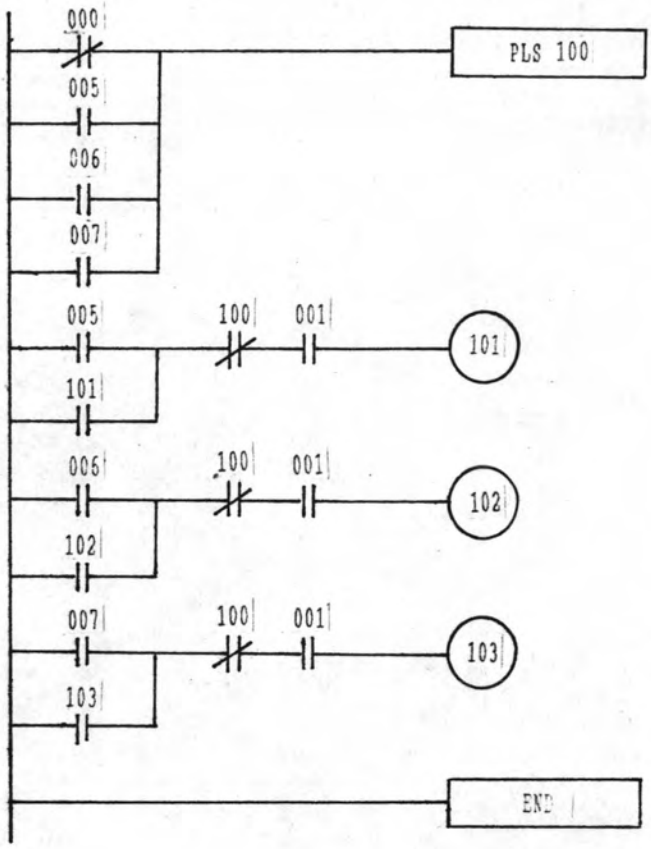
3.3.1 คำสั่งเบื้องต้น (Basic Instruction) ของ PC

จะประกอบด้วยคำสั่ง นีมอนิค (Mnemonic) โอเปอแรนด์ (Operand) และสัญลักษณ์ดังแสดงในรูปที่ 3.2

จากคำสั่งเบื้องต้นเหล่านี้สามารถนำไปใช้เขียนโปรแกรมนีมอนิค (MNEMONIC) แทนวงจรขั้นบันไดได้ ดังแสดงในรูปที่ 3.3 และรูปที่ 3.4 โปรแกรมที่แปลงแล้วจะถูกเก็บไว้ในหน่วยความจำของ PC ในการประมวลผลของโปรแกรมนีมอนิค (MNEMONIC) นี้ ซีพียูจะทำการอ่านคำสั่งในโปรแกรมเพื่อนำมาแปลความหมายแล้วทำตามคำสั่งนั้น ๆ โปรแกรมควบคุมการแปลความหมายนี้เรียกว่า Ladder Interpreter ซึ่งมีขั้นตอนการทำงานดังแสดงในรูปที่ 3.5

คำสั่ง Instruction	สัญญลักษณ์ Symbol	นีมอนิค Mnemonic	โอเปอแรนด์ Operand
LOAD		LD	Relay No.
LOAD-NOT		LD NOT	Relay No.
AND		AND	Relay No.
AND-NOT		AND NOT	Relay No.
OR		OR	Relay No.
OR-NOT		OR NOT	Relay No.
AND-BLOCK		AND BLK	
OR-BLOCK		OR BLK	
OUT		OUT	Relay No.
DATA SET		DS	Set value
TIMER		TIM DS	Timer No. Set value
COUNTER		CNT DS	Counter No. Set value
SHIFT		SFT	CH-No.
PULSE		PLS	Relay No.
END		END	

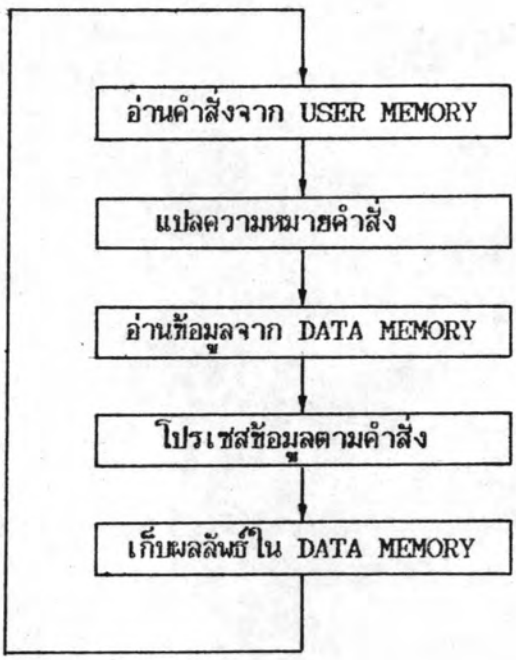
รูปที่ 3.2 แสดงคำสั่งเบื้องต้นที่ใช้ใน PC



รูปที่ 3.3 แสดงตัวอย่างโปรแกรมขั้นบันไดที่ใช้ควบคุม

STEP	MNEMONIC	OPERAND
000	LD-NOT	000
001	OR	005
002	OR	006
003	OR	007
004	PLS	100
005	LD	005
006	OR	101
007	AND-NOT	100
008	AND	001
009	OUT	101
010	LD	006
011	OR	102
012	AND-NOT	100
013	AND	001
014	OUT	102
015	LD	007
016	OR	103
017	AND-NOT	100
018	AND	001
019	OUT	103
020	END	-

รูปที่ 3.4 แสดงตัวอย่างโปรแกรมนีมอนิคที่เก็บในหน่วยความจำผู้ใช้



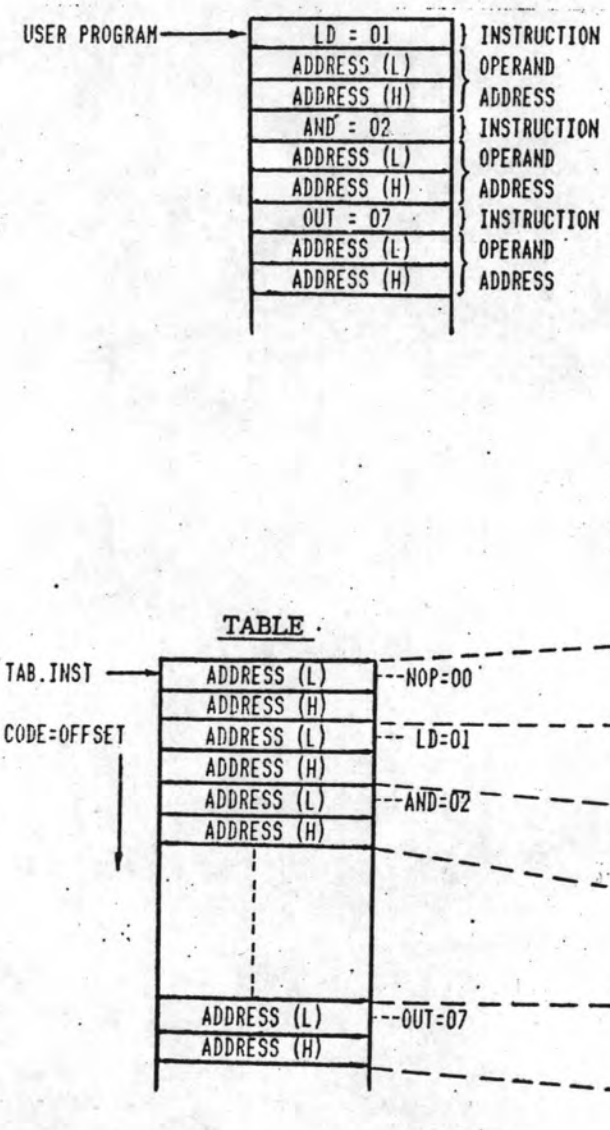
รูปที่ 3.5 แสดงวงรอบการทำงานของตัวแปลคำสั่ง

3.3.2 เทคนิคของโปรแกรมควบคุมการแปลคำสั่ง (Ladder Interpreter)

ส่วนสำคัญของโปรแกรมอยู่ที่วิธีการและรูปแบบของการเก็บโปรแกรมนีมอนิคของวงจรขั้นบันได (Ladder Diagram) ที่แปลงแล้วในหน่วยความจำเพื่อการประมวลผลซึ่งการเก็บโปรแกรมนีมอนิคที่คิดขึ้นมามีอยู่ด้วยกันหลายวิธี แต่ละเทคนิคมีความเร็วในการประมวลผลและเวลาการค้นหาขั้นของโปรแกรมยาวไม่เท่ากัน

เทคนิคของโปรแกรมควบคุมการแปลคำสั่งแต่ละแบบมีหลักการสำคัญดังต่อไปนี้

1. **วิธี TABLE LOOK UP** การประมวลผลด้วยวิธีนี้จะมีรูปแบบการเก็บคำสั่งของโปรแกรมผู้ใช้และวิธีการประมวลผล ดังแสดงในรูปที่ 3.6

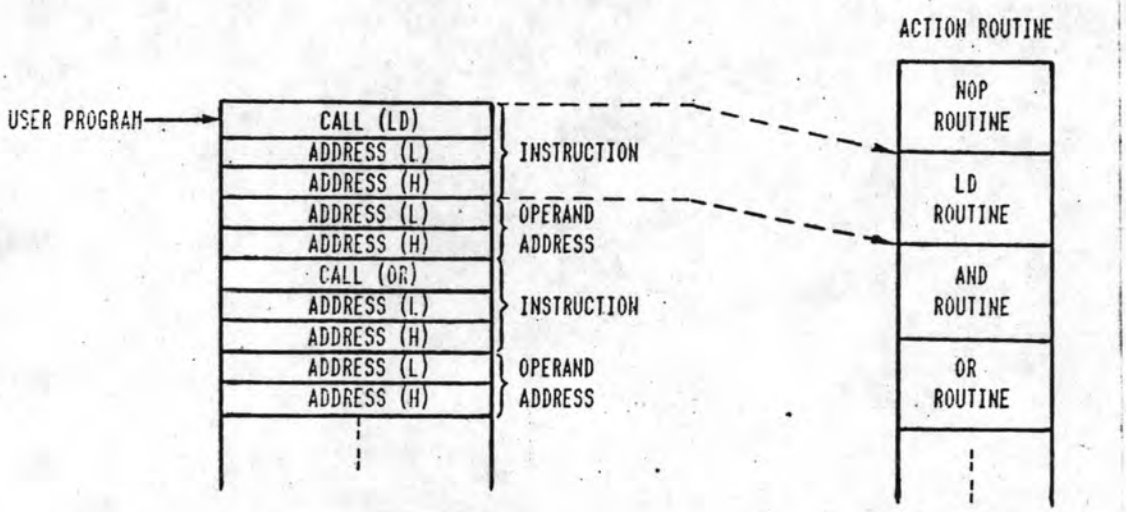


รูปที่ 3.6 แสดงการแปลคำสั่งด้วยวิธี TABLE LOOK UP

วิธีการเก็บคำสั่ง และ OPERAND ทำได้โดยการเก็บ โค๊ด ของคำสั่งใน Byte แรก และตามด้วย OPERAND ADDRESS ในอีก 2 Byte ถัดไป ดังนั้นวิธีนี้จะใช้เนื้อที่ในการเก็บคำสั่ง 3 Byte/Step ในการประมวลผลคำสั่งจะต้องนำ โค๊ด ในแต่ละขั้นโปรแกรม (Step)

ที่เก็บใน โปรแกรมผู้ใช้ (User Memory) ซึ่งเป็น โค๊ด ของคำสั่ง เช่น คำสั่ง LD = 01, OUT = 07 เป็นต้น โค๊ด ที่เก็บนี้จะนำมาใช้เป็น OFFSET เพื่อนำไปใช้ในการเปิดตาราง (Table) เพื่อหาแอดเดรส ของ ACTION ROUTINE ที่ต้องการ เมื่อพบแล้วก็จะประมวลผลตาม ACTION ROUTINE นั้น ๆ โดยในแต่ละ ACTION ROUTINE จะดึงเอา OPERAND ADDRESS ที่อยู่ในโปรแกรมผู้ใช้มาใช้เพื่อหาสภาวะการทำงานของรีเลย์ภายใน ความเร็วในการประมวลผลด้วยวิธีนี้ค่อนข้างช้า เพราะเสียเวลาในการเปิดตาราง (Table)

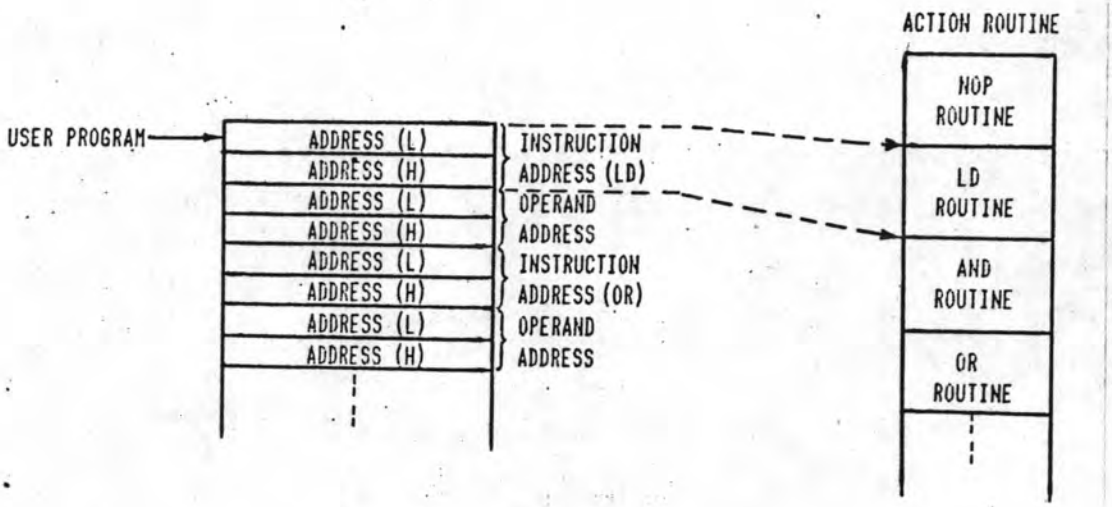
2. **วิธี CALL** การประมวลผลด้วยวิธีจะมีรูปแบบการเก็บคำสั่งของโปรแกรมผู้ใช้ และวิธีการประมวลผลดังแสดงในรูปที่ 3.7



รูปที่ 3.7 แสดงการแปลคำสั่งด้วยวิธี CALL

วิธีการเก็บคำสั่ง และ OPERAND ทำได้โดยการเก็บคำสั่ง CALL ที่มี SUBROUTINE เป็น ACTION ROUTINE ของคำสั่งที่ต้องการจะประมวลผล ใช้เนื้อที่ในการเก็บคำสั่ง 3 Byte และตามด้วย OPERAND ADDRESS อีก 2 Byte ดังนั้นวิธีนี้จึงใช้เนื้อที่ในการเก็บคำสั่ง 5 Byte/Step แต่มีข้อดีที่การประมวลผลของคำสั่งจะ CALL ไปยัง ACTION ROUTINE ทันที โดยไม่ต้องเสียเวลาในการเปิดตารางเหมือนวิธีแรก โปรแกรมภายใน ACTION ROUTINE จะดึง OPERAND ADDRESS ที่เก็บไว้ใน โปรแกรมผู้ใช้ มาใช้เช่นเดียวกับที่กล่าวมาแล้ว

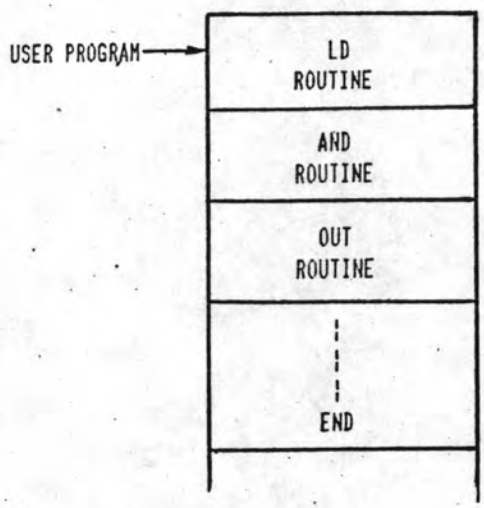
3. วิธี AUTO INCREMENT การประมวลผลโดยวิธีนี้จะมีรูปแบบการเก็บคำสั่งของ โปรแกรมผู้ใช้ และมีวิธีประมวลผลดังแสดงในรูปที่ 3.8



รูปที่ 3.8 แสดงการแปลคำสั่งด้วยวิธี AUTO INCREMENT

การเก็บคำสั่ง และ OPERAND ของโปรแกรมผู้ใช้ โดยเก็บ INSTRUCTION ADDRESS ของ ACTION ROUTINE 2 Byte และตามด้วย OPERAND อีก 2 Byte วิธีนี้ใช้เนื้อที่ในการเก็บคำสั่ง 4 Byte/Step ส่วนวิธีประมวลผลจะใช้ USER MEMORY เป็นเสมือน STACK โดยให้ STACK POINTER ชี้ไปยัง โปรแกรมผู้ใช้ ดังนั้นในการประมวลผลคำสั่งจะใช้คำสั่ง POP H เพื่ออ่านแอดเดรสของคำสั่ง มาเก็บไว้ในรีจีสเตอร์ HL และใช้คำสั่ง PCHL เพื่อโหลดแอดเดรสนั้นเข้าไปใน PROGRAM COUNTER เป็นการ JUMP ไปยัง ACTION ROUNTINE ที่ต้องการ ในการใช้คำสั่ง POP แต่ละครั้ง PROGRAM COUNTER จะเพิ่มค่าและชี้มายังแอดเดรสถัดไปโดยอัตโนมัติ วิธีนี้ทำให้ความเร็วในการประมวลผลเร็วขึ้นกว่า 2 วิธีที่กล่าวมาข้างต้น

4. วิธี COMPILER การประมวลผลด้วยวิธีนี้จะมีรูปแบบการเก็บคำสั่งของโปรแกรมผู้ใช้และมีวิธีประมวลผลดังแสดงในรูปที่ 3.9



รูปที่ 3.9 แสดงการแปลคำสั่งด้วยวิธี COMPILER

การเก็บคำสั่ง และ OPERAND ด้วยวิธีนี้จะเก็บ ACTION ROUTINE ของแต่ละขั้นโปรแกรมในโปรแกรมผู้ใช้ วิธีนี้ใช้เนื้อที่ในการเก็บ Byte/Step ไม่แน่นอนขึ้นอยู่กับความยาวของแต่ละคำสั่ง วิธีนี้จะให้ความเร็วในการประมวลผลดีที่สุด

จากวิธีการประมวลผลที่กล่าวมาแล้วทั้ง 4 วิธี วิธีที่ 3 และวิธีที่ 4 จะให้ความเร็วในการประมวลผลดีกว่าวิธีที่ 1 และวิธีที่ 2 ในการทำวิทยานิพนธ์นี้จะใช้การประมวลผลวิธีที่ 3 (วิธี AUTO INCREMENT) เนื่องจากวิธีนี้มีโครงสร้างในการเก็บโปรแกรมของแต่ละขั้นโปรแกรมยาวคงที่ ทำให้ง่ายต่อการแก้ไข และค้นหาโปรแกรมในแต่ละขั้นโปรแกรม เพื่อมอนิเตอร์ดูสภาวะการทำงานของรีเลย์ภายใน นอกจากนี้ยังมีโครงสร้างของข้อมูลที่ไม่ยุ่งยาก ดังในตารางที่ 3.1 แสดงการเปรียบเทียบโปรแกรมควบคุมการแปลคำสั่ง (Ladder Interpreter) ต่าง ๆ ที่ได้กล่าวมาแล้ว

ตารางที่ 3.1 แสดงการเปรียบเทียบโปรแกรมควบคุมการแปลคำสั่ง (Ladder Interpreter)

	TABLE look up	วิธี CALL	AUTO INCREMENT	วิธี COMPILER
BYTE/STEP	3	5	4	4 - 15
EXECUTION SPEED (APPROX) µS/STEP	200	115	55	20
กระบวนการแปลงระหัสและถอดระหัส	ง่าย	ปานกลาง	ปานกลาง	ยาก
การจัด DATA STRUCTURE	ง่าย	ปานกลาง	ปานกลาง	ยาก

หมายเหตุ EXECUTION SPEED เปรียบเทียบที่ CLOCK 330 NSEC.