ปีการศึกษา 2532

ปริญญานิพนธ์เรื่อง

PROGRAMMABLE LOGIC CONTROLLER

โดย

นาย สมชาติ แซ่อึ้ง

นาย สมหวัง เจริญยงอยู่

อาจารย์ที่ปรึกษา

อ. สุเธียร เกียรสุนทร

คณะวิศวกรรมศาสตร์ ภาควิชา ระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

PROGRAMMABLE LOGIC CONTROLLER

สมชาติ แซ่อึ้ง 291246

สมหวัง เจริญยงอยู่ 291283

อาจารย์ที่ปรึกษา

...สุธิธร...เกียรติสุนทร.....

อ. สุธิธร เกียรติสุนทร

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการจำลอง PLC โดยใช้ Z-80 Microprocessor และโปรแกรมการใช้งาน ซึ่งจะถูกส่งจากคอมพิวเตอร์ ผ่านทาง Series port โดยใช้ RS-232 Series Communication และการสั่งการจากเครื่องคอมพิวเตอร์

ABSTACT

This project use Z-80 microprocessor as PLC (programmable logic controler) which transfer control program and user program by personal computer using RS-232 serial communication

สารบัญ

บทที่ 1
บทนำ

PLC (PROGRAMMABLE LOGIC CONTROLLER) จะทำงานแบบ on และ OFF ใช้สำหรับควบคุมขบวนการต่างๆ ทำให้งานตามเป้าหมายที่ตั้งไว้ สำหรับ PLC จะใช้ Z80ไมโครโปรเซสเซอร์เป็น CPU ของเครื่องควบคุม โดยโครงสร้างทั่วไปจะมีหน่วยประมวลผล หน่วย INPUT หน่วย OUTPUT และหน่วยป้อนโปรแกรม

จากโครงสร้างนี้โดยปกติหน่วยป้อนโปรแกรมจะเป็น KEYBOARD ซึ่งจะติดตั้งถาวรบนเครื่องเลย หรือสามารถถอดประกอบได้ แต่เป็นการป้อนโปรแกรมโดยใช้บุคคลากรเข้าไปป้อนโปรมแกรมที่เครื่อง PLC

จะเห็นได้ว่าถ้าขบวนการนั้นมีการเปลี่ยนแปลงขั้นตอนบ่อยๆ หรือสลับไปมาระหว่างขบวนการหลายขบวนการ ดังนั้นจะต้องมีโปรแกรม PLC สำหรับแต่ละขบวนการ และจะต้องป้อนโปรแกรมใหม่ทุกครั้ง

ในโครงงานนี้จึงจะใช้การควบคุมเครื่อง PLC โดยใช้เครื่อง COMPUTER ในการควบคุม จะสั่งการจากเครื่อง COMPUTER ให้ส่งโปรแกรม PLC ไปให้เครื่องPLC จัดการ และจากการใช้เครื่อง COMPUTER นี้เราจึงเก็บได้โปรแกรม PLC ได้หลายโปรแกรมและในการใช้งานครั้งต่อไปไม่ต้องทำการ KEY โปรแกรมใหม่

เนื้อหาและทฤษฎีต่าง ๆ ซึ่งได้แจกแจงรายละเอียด ดังนี้

ในบทที่ 1 ได้กล่าวที่มา และส่วนประกอบต่าง ๆ ของโครงงานนี้

บทที่ 2 อธิบายถึง ทฤษฎีเบื้องต้น และโครงสร้างของ PLC ทั่ว ๆ ไป และภาษาที่ใช้ในการโปรแกรมคำสั่ง

บทที่ 3 กล่าวถึง คำสั่งภาษาที่ใช้ในโครงงานนี้รวมทั้งตัวอย่างในการใช้งาน และจะกล่าวถึงโครงสร้างทางฮาร์ดแวร์และการออกแบบ

บทที่ 4 จะกล่าวถึง โปรแกรมบนส่วนของ Single Board

บทที่ 5 จะกล่าวถึง ส่วนประกอบของโปรแกรม Pascal ที่ใช้จัดการต่าง ๆ และ Flow chart

บทที่ 6 สรุปผล วิจารณ์ และความคิดเห็นเพิ่มเติม

บทที่ 2

ทฤษฎีเบื้องต้นและโครงสร้าง PLC

2.1 ความเป็นมา

PLC เป็นเครื่องมือควบคุมที่สร้างขึ้นมา เพื่อที่จะให้ใช้งานในระบบควบคุมแบบลำดับ อย่างต่อเนื่องได้ เครื่องควบคุมแบบนี้สามารถกำหนดโปรแกรมการทำงานได้ เพื่อให้ระบบทำงานตามช่วงเวลา ตามลำดับขั้นการทำงานและตามเงื่อนไขที่กำหนดไว้ สถาปัตยกรรมของ PLC ได้ถูกพัฒนาให้มีขีดความสามารถสูงขึ้นและสามารถเชื่อมต่อกับเครื่องคอมพิวเตอร์ได้ เพื่อวัตถุประสงค์อื่นๆ ได้ เช่น การเก็บข้อมูล การกระจายการควบคุม เป็นต้น ดังนั้นสามารถที่จะสั่งการควบคุมจากศูนย์คอมพิวเตอร์

2.2 ทฤษฎีเบื้องต้น

ในโครงงานนี้จะมีทฤษฎีและความรู้ที่จำเป็นต่างๆ ดังนี้

2.2.1 ระบบเลขฐานสอง และระบบ Digital logic

PLC เป็นเครื่องควบคุมที่ใช้การควบคุมแบบ logic และให้ผลออกมาเป็นแบบ ON - OFF ซึ่งจะใช้แทนรีเลย์ (RELAY) คือจะมี 2 สภาวะคือ ON จะมี LOGIC HIGH หรือ 1 และ OFF จะมี LOGIC LOW หรือ 0 และในการควบคุมจะใช้ FUCNTION LOGIC เช่น AND OR OUT และ NOT ฯลฯ PLC เป็นอุปกรณ์ดิจิตอลที่สามารถโปรแกรมการทำงานได้ การนำพีแอลซีไปควบคุมเครื่องจักรหรืออุปกรณ์อื่นๆ นั้น PLC จะติต่อกับเครื่องจักรหรืออุปกรณ์ในรูปตัวแปรเลขฐานสอง โดย PLC จะนำตัวแปรอินพุท (INPUT) ที่ได้มาทำการประมวลผลตามวงจรควบคุมที่ต่อกันในลักษณะเดียวกันกับฟังก์ชั่นทั้งสามที่กล่าวมา ฟังก์ชั่นเหล่านี้ PLC สร้างขึ้นมาด้วยวิธีทางซอฟร์แวร์ (SOFTWARE) ผลลัพธ์ที่ได้จากการประมวลจะอยู่ในรูปตัวแปรเอาท์พุท (OUTPUT) ที่ PLC จะนำไปควบคุมอุปกรณ์เหล่านั้น ในที่นี้มาพิจารณาฟังก์ชั่นทั้งสามดังต่อไปนี้คือ

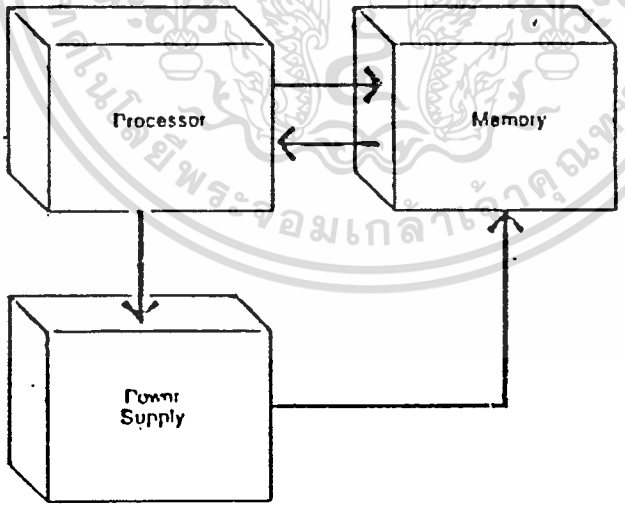
1. ฟังก์ชั่นแอนด์
2. ฟังก์ชั่นออร์
3. ฟังก์ชั่นน็อท

ไว้ในหน่วยความจำ (Memory) จากนั้นก็ส่งสัญญาณควบคุมเอาท์พุทออกไปเพื่อทำการควบคุมระบบต่อไป

- หน่วยประมวลผลกลางนี้ประกอบด้วยส่วนต่าง ๆ 3 ส่วน คือ

1. หน่วยประมวลผล (Processor)
2. หน่วยความจำ (Memory)
3. แหล่งจ่ายไฟ (Power supply)

รูปที่ 2-2 แสดงถึงโครงสร้างของหน่วยประมวลผลกลาง ในปัจจุบันหน่วยประมวลผลจะใช้ไมโครโปรเซสเซอร์ (Microlprocessor) ซึ่งมีขนาดเล็ก แต่มีความสามารถในการคำนวณทางคณิตศาสตร์และทางลอจิก โดยมีความเร็วในการคำนวณสูง และยังสามารถควบคุมระบบที่มีการตัดสินใจทำให้ระบบดังกล่าวมีประสิทธิภาพสูงขึ้น



รูปที่ 2-2 โครงสร้างของหน่วยประมวลผลกลาง

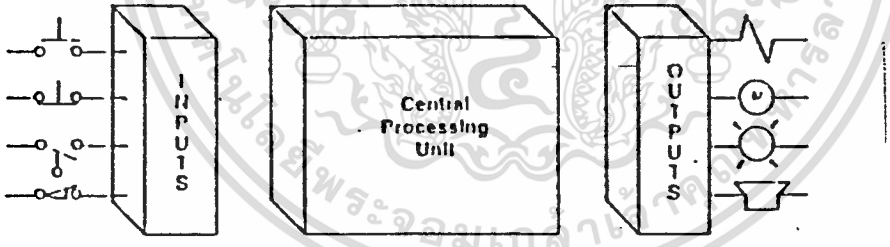
รายละเอียดของฟังก์ชั่นเหล่านี้ สามารถจะหาอ่านได้ตามหนังสือวงจรดิจิตอลทั่ว ๆ ไป

2.3 โครงสร้างของ PLC

โครงสร้างของ PLC ประกอบด้วยส่วนสำคัญ 2 ส่วน คือ

1. หน่วยประมวลผลกลาง หรือ ซีพียู (CPU : Central Processing Unit)
2. หน่วยอินพุท/เอาท์พุท (Input/Output Unit)

นอกจากส่วนประกอบทั้งสองแล้ว PLC ยังประกอบด้วยหน่วยป้อนโปรแกรม (Programming Unit) ซึ่งมีไว้สำหรับให้ผู้ใช้สามารถติดต่อกับ PLC ลักษณะการติดต่อนี้อาจเป็นการป้อนโปรแกรมควบคุม การแก้ไขโปรแกรม ตลอดจนตรวจสอบสภาพการทำงานของ PLC รูปที่ 2 - 1 แสดงถึงโครงสร้างของ PLC



รูปที่ 2 - 1 โครงสร้างของ PLC

2.4 หน่วยประมวลผลกลาง (CPU)

หน่วยประมวลผลกลางเป็นหน่วยที่ทำหน้าที่ประมวลผลของ PLC โดยจะรับข้อมูลอินพุทจากอุปกรณ์ตรวจวัด (Sensing device) แล้วทำการประมวลผลตามโปรแกรมที่เก็บ

2.4.1 หน่วยประมวลผล (Processor)

หน่วยประมวลผลทำหน้าที่ควบคุมการทำงานของระบบทั้งหมด โดยรับข้อมูลอินพุทเข้ามาทำการประมวลผลแล้วส่งผลที่ได้ออกไป จากนั้นก็จะวนกลับไปรับข้อมูลอินพุทเข้ามาอีก และจะทำซ้ำ ๆ กันในลักษณะวนรอบเช่นนี้เรื่อย ๆ ไป ในการทำงานแต่ละรอบนี้เรียกว่าการสแกน (Scaning) เวลาของการสแกนขึ้นกับขนาดของหน่วยความจำ และความเร็วของหน่วยประมวลผล ช่วงเวลาของการสแกนจะทำให้ทราบถึงความสามารถในการตอบสนองต่อการเปลี่ยนแปลงของอินพุท/เอาท์พุทของ PLC ว่ามีความรวดเร็วเพียงใด รูปที่ 2-3 แสดงถึงลักษณะการสแกนของ PLC ที่มีการตอบสนองต่ออินพุท/เอาท์พุท

2.4.2 หน่วยความจำ (Memory)

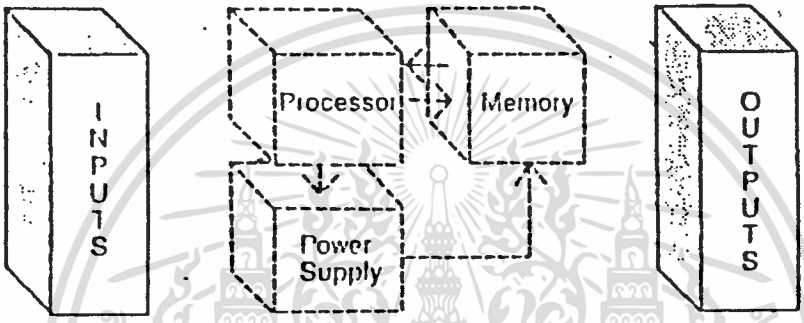
หน่วยความจำเป็นส่วนหนึ่งของซีพียู ใช้สำหรับเก็บโปรแกรมควบคุมระบบที่ป้อนโดยผู้ใช้ และโปรแกรมมอนิเตอร์ (Monitor Program) ของตัว PLC เอง อุปกรณ์ที่ใช้เป็นส่วนความจำคือแรม (RAM : Ramdom Access Memory) และอีพรอม (EPROM : Erasable Programmable Read Only Memory) โดยแรมเป็นส่วนความจำที่ใช้เก็บโปรแกรมควบคุมที่ป้อนโดยผู้ใช้ ทั้งนี้เพราะโปรแกรมควบคุมนี้อาจต้องมีการเปลี่ยนแปลงแก้ไข ดังนั้นจึงจำเป็นต้องใช้ส่วนความจำที่สามารถลบข้อมูลเดิม และนำโปรแกรมใหม่เข้าไปเก็บไว้ได้ ในการใช้งานจริง ๆ แล้วต้องมีแหล่งจ่ายไฟสำรองเพื่อป้องกันไม่ให้ข้อมูลสูญหายเมื่อไฟฟ้าดับ ส่วนความจำอีกอันหนึ่งคืออีดรอมซึ่งจะใช้เก็บโปรแกรมมอนิเตอร์ของ PLC โปรแกรมนี้จะทำหน้าที่ควบคุมการรับข้อมูล หรือโปรแกรมที่ป้อนให้กับเครื่อง PLC โดยอำนวยความสะดวกในการใช้งานให้ผู้ใช้สามารถใช้เครื่องได้ง่ายและยังทำหน้าที่ควบคุมการประมวลผลอีกด้วย

2.4.3 แหล่งจ่ายไฟ (Power Supply)

แหล่งจ่ายไฟเป็นส่วนที่ทำหน้าที่จ่ายแรงดันไฟฟ้าให้กับหน่วนประมวลผล หน่วยความจำ และหน่วยอินพุท/เอาท์พุท ตามที่ต้องการ แหล่งจ่ายไฟยังทำหน้าที่รักษาระดับแรงดันไฟฟ้าที่จ่ายให้ได้ตามต้องการด้วย

2.5 หน่วยอินพุท/เอาท์พุท

หน่วยอินพุ/เอาท์พุท เป็นตัวกลางเชื่อมการติดต่อระหว่างหน่วยประมวลผลกลางกับอุปกรณ์ภานนอกโดยที่หน่วยอินพุททำหน้าที่รับค่าสภาวะหรือปริมาณทางกายต่าง ๆ จากอุปกรณ์ตรวจวัดของเครื่องจักรหรือกระบวนการ เช่น ลิมิตสวิทช์ (Limit Switch) พร็อกซิมิตีสวิทช์ (Proximity Switch) และอื่น ๆ ส่งไปยังหน่วยประมวลผลกลางเพื่อประมวลผลตามขั้นตอนของโปรแกรมของผู้ใช้



รูปที่ 2-4 หน่วยอินพุท/เอาท์พุท

หน่วยอินพุท/เอาท์พุท ทำหน้าที่รับค่าสภาวะที่ได้จากการประมวลผลของหน่วยประมวลผลกลาง เพื่อนำค่านี้ไปควบคุมอุปกรณ์ของเครื่องจักรหรือกระบวนการต่าง ๆ เช่น มอเตอร์ (Motor) วาล์ว (Valve) โซลินอยล์ (Solenoid) และอื่น ๆ เป็นต้น

2.6 หน่วยป้อนโปรแกรม

หน่วยป้อนโปรแกรมของ PLC/PC ทำหน้าที่ติต่อระหว่างผู้ใช้ และ PLC/PC โดยที่ผู้ใช้ทำการป้อนโปรแกรมคำสั่งเข้าสู่หน่วยความจำของ CPU เพื่อเป็นการตรวจสอบสภาพการทำงาน

1. เครื่องป้อนโปรแกรมแบบ CRT ประกอบด้วยจอภาพและแป้นพิมพ์ เป็นแบบที่สะดวกที่สุด แบ่งออกเป็น 2 ชนิดคือ แบบ Dump CRT ซึ่งเป็นแบบที่ไม่มีหน่วยประมวลผลอยู่ภานในตัว การทำงานทุกอย่งจะถูกควบคุมจาก CPU ของ PLC/PC และแบบ Intelligent CRT ซึ่งจะมีหน่วยประมวลผลอยู่ภานใน ตัวการทำงานเป็นอิสระจาก CPU จึงมีประสิทธิภาพดีกว่าแบบแรก

2. เครื่องป้อนโปรแกรมขนาดเล็ก (Mini Programmer) มีขนาดเล็กสามารถเคลื่อนย้ายไปมาได้สะดวก ใช้ LED หรือ LCD ในการแสดงผล เครื่องป้อนโปรแกรมแบบนี้แบ่งออกได้เป็น 2 ชนิด คือ Dump และ Intelligent เช่นเดียวกัน แต่ประสิทธิภาพจะด้อยกว่าเครื่องป้อนโปรแกรมแบบ

3. เครื่องป้อนโปรแกรมลงในหน่วยความจำของ PLC/PC หรือเทป เครื่องป้อนโปรแกรมแบบนี้สามารถใช้ป้อนหรือแก้ไขโปรแกรมคำสั่งของผู้ใช้ เข้าไปในหน่วยความจำของ 1/4LC/PC ได้โดยตรง หรือใช้ในการับส่งโปรแกรม และข้อมูลต่าง ๆ ระหว่าง PLC/PC และเทป

4. เครื่องป้อนโปรแกรมลงในหน่วยความจำ (Memory Burner) เครื่องป้อนโปรแกรมแบบนี้ทำหน้าที่ถ่ายโปรแกรมที่มีอยู่ลงเก็บไว้ในหน่วยความจำแบบ ROM เพื่อให้โปรแกรมต่าง ๆ คงอยู่ตลอดเวลาเมื่อโปรแกรมต่าง ๆ ได้ถูกตรวจสอบ และแก้ไขจนเรียบร้อยสมบูรณ์แล้ว

5. คอมพิวเตอร์ PLC/PC บางเครื่องออกแบบมาเพื่อให้ใช้ติดต่อกับเครื่องคอมพิวเตอร์ได้ ทำให้สามารถเขียนโปรแกรมทุกอย่างโดยใช้คอมพิวเตอร์

ในโครงการนี้ใช้ติดต่อกับเครื่องคอมพิวเตอร์ PC/Xt ทำงานส่งโปรแกรมลงบน Single Board

2.7 ภาษาที่ใช้

PLC/PC เป็นเครื่องมือควบคุมที่ใช้วิธีการเขียนโปรแกรมและใช้ซอฟแวร์แทนอุปกรณ์ต่าง ๆ ดังนั้น การเขียนโปรแกรมต่าง ๆ จึงใช้สัญญลักษณ์พื้นฐานต่อไปนี้แทนอุปกรณ์ต่าง ๆ

+-----][-----+ 1. หน้าสัมผัสปกติเปิดหรือ N.O. ใช้แทนอุปกรณ์ทางด้านอินพุตต่าง ๆ ได้แก่ สวิทช์ไฟฟ้า หน้าสัมผัสของรีเลย์ตัวตั้งเวลาตัวนับ หรือรีเลย์ภายใน โดยที่อุปกรณ์นั้นปกติจะมีค่าสภาวะเป็น "Ø"

+-----][-----+ 2. หน้าสัมผัสปกติปิดหรือ N.C. ใช้แทนอุปกรณ์ทางด้านอินพุทต่าง ๆ โดยที่อุปกร์นั้นปกติจะมีค่าสภาวะเป็น "1"

+-----()-----+ 3. เอาท์พุทปกติไม่ทำงาน ใช้แทนอุปกรณ์ทางด้านเอาท์พุทต่าง ได้แก่หลอดไฟฟ้า มอเตอร์ไฟฟ้า ขดลวดของรีเลย์ ขดลวด Solenoid ตัวตั้งเวลา ตัวนับ รีเลย์ภายในโดยปกติแล้วทางด้านเอาท์พุทจะมีค่าสภาวะเป็น "ø"

+-----()-----+ 4. เอาท์พุทปกติทำงาน ใช้แทนอุปกรณ์ทางด้านเอาท์พุทต่างๆ ได้โดยปกติแล้วอุปกรณ์ทางด้านเอาท์พุทจะมีค่าสภาวะเป็น 1

ส่วนภาษาที่ใช้ในการเขียนโปรแกรมคำสั่งของ PLC/PC สามารถแบ่งออกได้เป็น 4 ภาษา ได้แก่

1. ภาษาแลดเดอร์ไดอะแกรม (Ladder Diagrams)
2. ภาษาคำสั่งบูลีน (Boolean Mnemonics)
3. ภาษาคำสั่งในรูปบล็อก (Functionals Blocks)
4. ภาษาระดับสูง (High Level Language)

สองภาษาแรกเป็นภาษาพื้นฐานของ PC เป็นภาษาที่เข้าใจง่าย เหมาะกับการเขียนคำสั่งในการควบคุมซีเควนซ์ที่เป็น ON/OFF ส่วนสองภาษาหลังจัดเป็นภาษาระดับสูง เหมาะสำหรับการควบคุมแบบอนาล็อค การจัดการเกี่ยวกับข้อมูล ฯลฯ

ภาษาแลดเดอร์ไดอะแกรม

ใช้สัญญลักษณ์ซึ่งคล้ายคลึงกับหน้าสัมผัสของรีเลย์ สัญญลักษณ์ที่ใช้ในการเขียนโปรแกรมคำสั่งได้แก่

สัญญลักษณ์	คำสั่ง
|----][----	LOAD
.----][----	AND
|----]/[---|	AND NOT
----( )----	OUT

ภาษาคำสั่งบูลีน

เป็นภาษาที่ใช้เขียนคำสั่งให้ PLC/PC ทำงานโดยใช้คำสั่งพื้นฐานทางลอจิก ร่วมกับชื่อสัญญลักษณ์ของคำสั่งอื่น ๆ ภาษานี้จะใช้กับ PC ขนาดเล็กและขนาดกลาง มีคำสั่งมาตรฐานบางคำสั่งดังนี้

สัญญลักษณ์	คำสั่ง
LD/STR	LOAD/START
AND	AND
OR	OR
NOT	NOT
OUT	OUT
TIM	TIMER
CNT	COUNTER
END	END

และอื่น ๆ

ภาษาระดับสูง

ภาษาระดับสูงของ PC เป็นการเขียนโปรแกรมตามคำสั่งด้วยภาษาที่คล้ายคลึงกับการโปรแกรมคอมพิวเตอร์ทั่วๆไป เช่น ภาษาเบสิค เป็นต้น ปกติแล้วจะใช้กับ PC ขนาดใหญ่ ซึ่งมีการทำงานที่ยุ่งยาก ซับซ้อน และมีเงื่อนไขขั้นตอนการตัดสินใจมาก

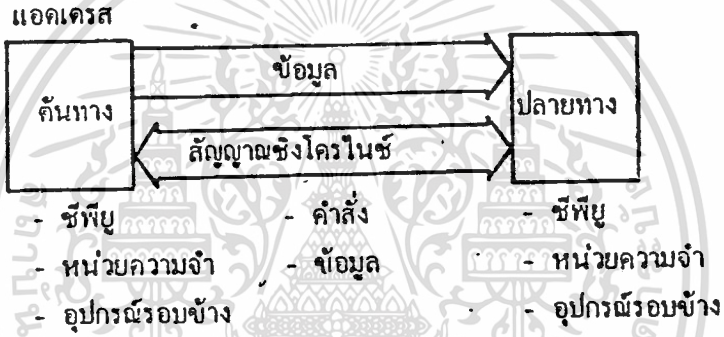
2.8 การส่งผ่านข้อมูล

การอินเตอร์เฟสไมโครโปรเซสเซอร์

ดังที่ทราบมาแล้วว่าขบวนการทางคณิตศาสตร์และลอจิกของระบบคอมพิวเตอร์เกิดขึ้นจากการทำงานของหน่วยประมวลผลกลาง ซึ่งเป็นวงจรรวม(Integrated Circuit ; IC) หรือที่เรียกกันว่า ซีพียู (Central Processing Unit ; CPU) นั่นเอง โดยไอซีตัว

นี้จะติดตั้งอยู่บนแผ่นวงจรร่วมกับอุปกรณ์อื่น ๆ เช่น หน่วยความจำ ROM (Read Only Memory) , Ram (Random Access Memory) และอุปกรณ์อื่น ๆ ดังนั้นการอินเตอร์เฟสไมโครโปรเซสเซอร์ก็คือ การทำงานร่วมกันระหว่าง ซีพียู กับอุปกรณ์ต่าง ๆ บนแผ่นวงจรนั่นเอง

นอกจาก ซีพียู จะต้องทำงานสอดคล้องกับ ROM , RAM แล้วยังต้องอินเตอร์เฟสเข้ากับอุปกรณ์อินพุท , เอาท์พุท ต่าง ๆ อีกด้วย เพื่อเพิ่มประสิทธิภาพการทำงานให้สมบูรณ์ยิ่งขึ้น ในขบวนการต่าง ๆ ของการทำงานร่วมกันของอุปกรณ์อิเล็กทรอนิกส์ จะต่อเนื่องกันเป็นลูกโซ่ เช่นในการส่งข้อมูลจากซีพียูไปยังอุปกรณ์อื่น เป็นต้น



รูปที่ 2.5 การอินเตอร์เฟสคอมพิวเตอร์

การส่งข้อมูลแบบขนานและแบบอนุกรม

โดยทั่ว ๆ ไปหลักใหญ่ของการส่งข้อมูลในคอมพิวเตอร์ หรือระหว่างคอมพิวเตอร์ด้วยกันจะมีลักษณะของการส่งข้อมูลอยู่ 2 แบบ คือ ส่งแบบขนาน และส่งแบบอนุกรม ดังกล่าวมาแล้วว่าคำสั่งหรือข้อมูลอยู่ในรูปของบิท คือหลาย ๆ บิทประกอบกันเป็นคำ ๆ หนึ่ง (Word) หรือคำสั่งหนึ่ง ๆ ดังรูปที่ 2.6 ได้แสดงถึงกลุ่มของบิทที่มีการใช้งานในไมโครคอมพิวเตอร์ โดยในการกำหนดแอดเดรสของหน่วยความจำ ,หรือการเขียนคำสั่ง และขบวนการอื่น ๆ ล้วนแต่ต้องแปลงให้อยู่ในรูปของเลขศูนย์กับเลขหนึ่งเสมอ จึงจะทำให้ซีพียูรับรู้ และปฏิบัติตามได้ จึงได้มีการกำหนดลักษณะมาตรฐานของข้อมูลดังนี้

"ถ้าข้อมูลหนึ่งตัวเมื่อแปลงให้อยู่ในรูปของเลขฐานสองแล้ว ประกอบด้วย 4 บิท เราเรียกว่า 4 บิทไมโคร หรือ 1 นิบเบิล (Nybble)"

"และถ้าข้อมูลประกอบไปด้วยกลุ่มของบิทที่มี 8 บิท เราเรียกว่าเป็น 1 ไบท์ (Byte)" เป็นต้น แต่ในระบบอื่นอาจจะมี 16 บิท หรือ 32 บิท เป็น 1 ไบท์ก็ได้

เพราะฉะนั้นเมื่อเรารู้ลักษณะของข้อมูลแล้ว ต่อไปเราจะมาดูถึงเรื่องข้อแตกต่างระหว่างการส่งข้อมูลในแบบขนาน และอนุกรมว่าเป็นอย่างไร

การส่งข้อมูลแบบอนุกรม : ข้อมูลแต่ละบิทจะถูกส่งเรียงกันออกไปเป็นลำดับ แต่เนื่องกันที่ละบิท เช่น ถ้าข้อมูลเป็น 1010 : เลข 0 ทางขวามือสุดซึ่งเป็น LSB(Least Significant bit) ส่งออกไปก่อน ตามด้วยบิทที่สองคือ เลข 1 และบิทที่ 3 คือเลข 0 และบิทสุดท้ายคือ เลข 1 ซึ่งเป็นMSB(MostSingniticantbit) ตามลำดับ โดยส่งข้อมูลจะมีอยู่เพียงเส้นเดียวเท่านั้น

สำหรับในระบบไมโครคอมพิวเตอร์ขนาด 8 บิทนั้น ซีพียู Z-80 ถูกใช้แพร่หลายที่สุดในการใช้งาน ซีพียู Z-80 บัสข้อมูลทั้ง 8 เส้น จะต่ออยู่กับรีจิสเตอร์ภายในของ ซีพียู โดยหน้าที่ของซีพียูนี้จะทำหน้าที่ในการแปลคำสั่งที่รับเข้ามา แล้วเปลี่ยนให้อยู่ในรูปของสัญญาณลอจิกส่งออกไปสั่งงานให้ขาต่าง ๆ ของตัวซีพียู(Z-80) มีขาที่ติดอยู่กับตัวถังทั้งหมด 40 ขา ประกอบกันเรียกว่าแพ็กเกจปฏิบัติงานส่งสัญญาณออกไปให้อุปกรณ์ต่าง ๆ บนแผ่นวงจรพิมพ์ สำหรับโครงสร้างอื่น ๆ ของซีพียู Z-80 ก็ประกอบด้วยหน่วยความจำ หน่วยความจำของระบบนี้จะจัดเป็นกลุ่ม ๆ ละ 8 บิท การส่งข้อมูลผ่านพอร์ทอินพุท/เอาท์พุท จะส่งครั้งละ 8 บิท รีจิสเตอร์ภายในที่เป็นรีจิสเตอร์ใช้งานทั่วไปจะมีขนาด 8 บิท แต่ในโครงสร้างภายในสามารถใช้รีจิสเตอร์ 8 บิทสองตัวรวมเป็น 16 บิทได้ เป็นต้น

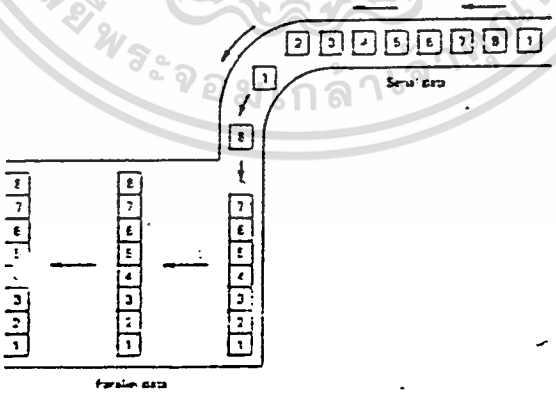
ในปัจจุบันไมโครคอมพิวเตอร์ขนาด 16 บิท ซุปเปอร์ไมโครคอมพิวเตอร์ ตลอดจนถึงมินิคอมพิวเตอร์ และซุปเปอร์มินิคอมพิวเตอร์ได้เข้ามามีบทบาทในงานคอมพิวเตอร์มา ดังนั้นหลักการต่าง ๆ จึงสามารถประยุกต์ใช้งานกับคอมพิวเตอร์เหล่านี้ได้

การส่งข้อมูลระหว่างซีพียูกับอุปกรณ์ต่าง ๆ บนแผ่นวงจรพิมพ์จะส่งเป็นแบบขนาน แต่ถ้าในกรณีที่เป็นการส่งข้อมูลจากไมโครคอมพิวเตอร์ไปยังอุปกรณ์รอบข้างต่าง ๆ การส่งข้อมูลจะเป็นการส่งแบบอนุกรมแทบทั้งสิ้น เพื่อเป็นการประหยัดค่าใช้จ่ายและสามารถส่งได้ระยะทางไกล ๆ ดังนั้นจึงใช้สายส่งข้อมูลเพียงเส้นเดียว ส่วนสายสัญญาณที่เหลืออยู่จะเป็นสายส่งสัญญาณควบคุม และสายกราวด์

ซึ่งมาตรฐานสากลที่กำหนดขึ้นมาควบคุมขบวนการส่งข้อมูลแบบนี้ประกอบด้วยมาตรฐานของสัญญาณระดับต่าง ๆ เช่น ระดับสัญญาณที่ใช้อุปกรณ์ TTL (Transistor-Transistor Logic), ระดับสัญญาณที่ใช้กับมาตรฐานของ EIA RS-232C และระดับสัญญาณที่ใช้กับระบบ 20 mA Current loop มาตรฐานสากลเหล่านี้ใช้กันแพร่หลายในการส่งข้อมูลแบบอนุกรมในระบบไมโครคอมพิวเตอร์โดยเฉพาะมาตรฐาน RS-232C

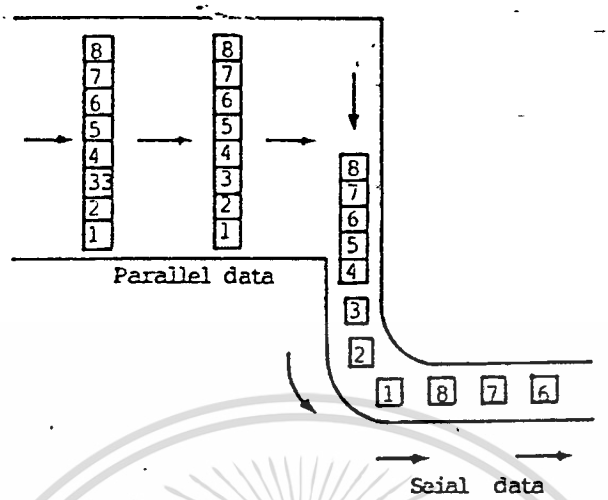
การแปลงรูปแบบข้อมูลแบบขนานไปเป็นแบบอนุกรม

หลักการทำงานของการแปลงรูปแบบข้อมูลนี้ก็โดยอาศัย ชิฟท์รีจิสเตอร์ โดยมีหลักการดังนี้ คือ ข้อมูลที่ส่งเข้ามาจะเป็นแบบอนุกรม โดยส่งเข้ามาทีละบิทเมื่อเข้ามาถึงรีจิสเตอร์ บิทแต่ละที่ทยอยกันเข้ามาก็จะถูกจัดเก็บเรียงกันอยู่ในรีจิสเตอร์ จนกระทั่งครบจำนวนบิทที่ต้องการ รีจิสเตอร์ก็จะส่งข้อมูลทั้งชุดนี้ออกไป ดังรูปที่ 2.8



รูปที่ 2.7 การแปลงรูปแบบข้อมูลจากแบบอนุกรมไปเป็นแบบขนาน

เมื่อต้องการจะแปลงรูปแบบข้อมูลกลับไปเป็นแบบอนุกรมอีก ก็ทำตามขบวนการที่ตรงกันข้ามดังรูปที่ 2.8



รูปที่ 2.8 การแปลงข้อมูลจากแบบขนานไปเป็นอนุกรม

ในการส่งข้อมูลลักษณะนี้จำเป็นที่จะต้องรู้จักคำ 3 คำต่อไปนี้ คือ

. UART : Universal Asynchronous Receiver-Transmitter.

. USART : Universal Synchronous Asynchronous Receiver Transmiiter

. SIO : Serial input/output circuit

ซึ่งอุปกรณ์เหล่านี้นอกจากจะต้องใช้งานร่วมกับซิฟรีจิสเตอร์แล้วยังมีประโยชน์ต่อระบบบัส , ความเร็วในการส่งข้อมูล , พาริตี้และฟอร์แมทข้อมูล เป็นต้น ตัวอย่างของวงจรรวมที่ใช้งานในลักษณะนี้ก็เช่น 8251 USART ซึ่งใช้งานร่วมกับซีพียู Z-80

การส่งข้อมูลแบบอะซิงค์โครนัสและซิงค์โคนัส

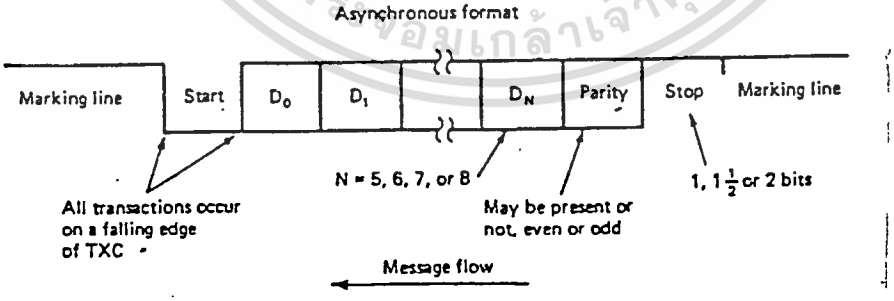
การรับส่งข้อมูลแบบซิโครนัส

ไม่ว่าในการส่งข้อมูลจะเป็นแบบขนานหรือแบบอนุกรม การส่งข้อมูลแบบซิงโครนัสก็คือระบบการส่งที่ข้อมูลแต่ละเวิร์ดออกไปตามเวลาที่แน่นอน ซึ่งหมายถึงระยะเวลาระหว่างข้อมูลแต่ละเวิร์ดที่ถูกส่งออกไปมีค่าแน่นอน

การรับส่งข้อมูลแบบอะซิงโครนัส

คือระบบการับส่งข้อมูลที่แต่ละคำ ถูกส่งออกไปอย่างไม่มีกำหนดเวลาแน่นอน นั่นคือ ระยะเวลาระหว่างข้อมูลแต่ละคำที่ถูกส่งออกไปมีค่าไม่แน่นอน ตัวอย่างเช่น การส่งข้อมูลจากคอมพิวเตอร์ A ไปยังคอมพิวเตอร์ B ในการป้อนข้อมูลจากคีย์บอร์ดนั้นผู้ที่พิมพ์ดีดไม่สามารถพิมพ์ด้วยอัตราเร็วคงที่ได้ แต่ละครั้งที่กดจะห่างไม่เท่ากัน ดังนั้นสิ่งที่กำหนดเวลาในการส่งข้อมูลก็คือ ความพร้อมเพียงของเครื่องรับและเครื่องส่ง

ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัสนั้น โครงสร้างของข้อมูลและบิทสุดท้ายคือบิทสิ้นสุดข้อมูล (Stop bit) โดยบิทเริ่มต้นจะแสดงถึงการเริ่มต้นของกลุ่มข้อมูล แล้วตามด้วยส่วนของกลุ่มข้อมูล และบางกรณีอาจจะมีการเพิ่มบิทพาริตี้เพื่อใช้ตรวจสอบความถูกต้องของข้อมูล และบิทสิ้นสุดข้อมูลในบล็อก ๆ นี้หมดลงแค่นี้ ส่วนในการส่งข้อมูลแบบซิงโครนัสนั้นจะต้องมีการส่งสัญญาณนาฬิกาไปพร้อม ๆ กับสัญญาณข้อมูล ในการส่งข้อมูลระยะสั้น ๆ สัญญาณนาฬิกาซึ่งใช้เป็นสัญญาณซิงค์ อาจจะส่งแยกไปในสายส่งสัญญาณอีกเส้นหนึ่งไม่ส่งรวมไปในสายส่งข้อมูลก็ได้แต่ถ้าเป็นการสื่อสารในระยะไกล ๆ แล้วสัญญาณนาฬิกาจะถูกเข้ารหัสส่งร่วมไปกับสัญญาณข้อมูลในสายส่งเส้นเดียวกัน และไม่ว่าจะเป็นการส่งข้อมูลแบบซิงโครนัสหรืออะซิงโครนัส ข้อมูลในบล็อกหนึ่ง ๆ จะต้องมีบิทที่แสดงให้รู้ว่าเป็นการเริ่มต้นของข้อมูล และสิ้นสุดข้อมูลซึ่งลักษณะของข้อมูลเช่นนี้เรากล่าวถึงมาแล้วในการส่งข้อมูลแบบอะซิงโครนัส ซึ่งแสดงได้ดังรูปที่ 2.11



รูปที่ 2.11 การส่งข้อมูลอนุกรมแบบอะซิงโครนัส

บทที่ 3

คำสั่งและการใช้งาน

สำหรับโครงงานนี้ได้กำหนดรูปแบบขึ้นมา , คำสั่งที่เขียนขึ้นจะเขียนขึ้นในลักษณะที่เข้าใจง่ายต่อการเขียน ทำให้ไม่ต้องมีการศึกษามาก ภาษาที่นิยมใช้มี 2 ภาษาคือ

1) ภาษาแลดเดอร์ไดอะแกรม (Leddre dirgram language)

2) ภาษาคำสั่งบูลลีน (Boolean language)

ในโครงการนี้ มีคำสั่งทั้งหมดคือ

LD numberbit;

เป็น คำสั่งที่ใช้ในการนำค่าบิทใดๆ จากอินพุท พอร์ตเข้ามาเก็บไว้ใน Accumulater บิทที่ ø และผลที่ได้เก็บไว้ใน stack เช่น LD 1; เป็นการนำค่าที่บิท 1 มาเก็บไว้

AND numberbit;

คำสั่งนี้เป็น คำสั่ง ถึงการต่ออนุกรมกันระหว่างผลลัพล์ทางลอจิกของขบวนการชุดคำสั่ง หรือโอเปอร์แรนด์ ก่อนหน้านี้ กับโอเปอร์แรนด์ ของคำสั่งนี้ เช่น

LD 1;

AND 2;

end

ลักษณะการทำงานของ Program นี้คือ นำค่า BIT ที่ 1 และ 2 มากระทำ AND กัน และเก็บไว้ที่ Stack

OR numberbit ;

คำสั่งนี้ แสดงถึงการต่อขนานกันระหว่างผลลับทางลอจิก ของขบวนการชุดคำสั่ง หรือ โอเปอร์แรนด์ ก่อนหน้านี้ กับโอเปอร์แรอนด์ของคำสั่งนี้ และผลที่ได้จากการออร์กันจะเก็บไว้ที่ สแตก

OUT numberbit ;

คำสั่งนี้ จะเป็นการนำค่าของการประมวลผลใน Stack มาออกทางเอาว์พุท พอร์ตตามที่ค่า numberbit กำหนด

XOR numberbit ;

คำสั่ง ช่วยในการกระทำ Xor กันระหว่างขบวนการชุดคำสั่งกอ่นหน้านี้ กับ โอ

เปอร์แรนด์ของคำสั่งนี้

... AND (...) ;

คำสั่งนี้จะเป็นการกระทำค่าประมวลผล AND กันระหว่าง โอเปอร์แรนด์ ภายใน ภายนอกลูปเป็นคำสั่งช่วยในการกระทำการ Load ซอ้มูล ซ้ำ แล้วเก้บไว้ใน สแตก ทด

... OR (...) ;

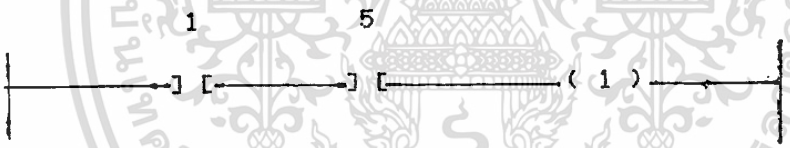
คำสั่งการกระทำค่าประมวลผล OR กันระหว่าง โอเปอร์แรนด์ ภายใน และ ภายนอกลูป

จากคำสั่งต่างๆ ข้างต้นนี้จะเป็นการกระทำอยู่ในลักษณะ make function ใต่การใช้งานแล้วจะมีการทำงานในลักษณะ Breake function อยู่ดวัย ดังนั้นจึงมีคำสั่งที่เหมาะสมสำหรับใช้งานในลักษณะนี้ คำสั่งเห่ลานี้ได้แก่ LDNOT , ANDNOT , ORNOT , และ OUTNOT จะหมายถึ่งการกระทำ Inverse ค่าบิทต่างๆ ที่กำหนด

END เป็นคำสั่งการจบโปรแกรม

ตัวอย่างการใช้คำสั่งเบื้องต้นต่าง ๆ

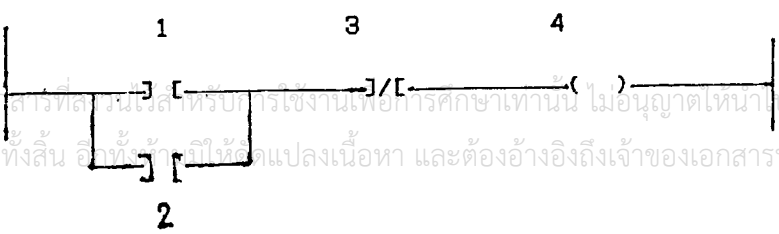
ตั่วอย่างที่ 1




```
โปรแกรม     LD    1 ;    เก็บบิทใน Accumulater
            AND   5 ;    ผลที่ได้นำมาอนุกรม กับ Input 5
            OUT   1 ;    การรวมกันทำงานมีผลต่อ การทำงานของ Output 1
            End
```

การทำงาน : Output ที่ bit 1 จะทำงาน (เป็น 1) เมื่อ ทั้ง bit 5 และ bit 1 ทำงาน (เป็น 1)

ตั่วอย่างที่ 2

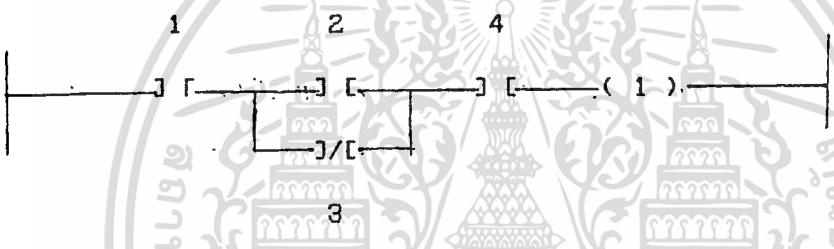


โปรแกรม

```
LD      1;
OR      2;
ANDNOT  3;
OUTNOT  4;
END
```

การทำงาน : OUTPUT ที่ bit 4 จะทำงานก็ต่อเมื่อ INPUT 1 หรือ INPUT 2 ทำงาน และ INPUT 3 ตอ้งไม่ทำงาน

ตัวอย่าง ที่ 3

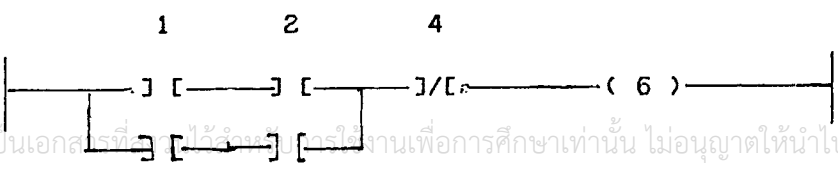


โปรแกรม

```
LD      1 ;
AND(    ;
LD      2 ;
ANDNOT  3);
AND     4 ;
OUT     1 ;
END
```

โปรแกรมนี้เป็นการนำ Register มาเก็บข้อมูลชั่วคราว เพื่อกลับมาใช้อีกภายหลัง

ตัวอย่างที่ 4



โปรแกรม

```
LD      1 ;
AND     2 ;   เก็บค่าไว้ใน  stack เดิม
OR(     ;
LD      5 ;   จะมีการเพิ่มค่า Stack address
AND   3) ;
ANDNOT  4 ;
OUT     b ;
```

การทำงาน : OUTPUT bit 6 จะทำงานก็ต่อเมื่อ บิท 1 และ บิท 2 ทำงาน และ บิท 4 เป็น OFF

บทที่ 4

การออกแบบและการสร้าง

การออกแบบได้แบ่งออกเป็น 2 ส่วน คือส่วน SOFEWARE และส่วนของ HARDWARE โดยส่วนของ SOFTWARE แบ่งออกเป็น 2 ส่วนคือ ส่วนที่เขียนบนเครื่องคอมพิวเตอร์กับส่วนที่เขียนบน SINGLE BOARD z-80 ส่วนของโปรแกรมบนคอมพิวเตอร์นั้นจะเป็นส่วนของการแปลงโค๊ด (CODE) และการตรวจสอบรูปแบบการเขียนโปรแกรม จากนั้นจะส่ง CODE มาบน RAM บน SINGLE BOARD รับในการประมวล ส่วนโปรแกรมบน SINGLE BOARD ก็จะประมวลผลของ CODE นั้น ๆ ตามลักษณะการเขียนโปรแกรม

ตามลักษณะทางด้าน HARDWARE

ลักษณะของเครื่อง

อินพุท 8 บิท และอินพุททาง SERIES PORT COMMUNICATION

เอาท์พุท 8 บิท

แหล่งจ่ายไฟ 5 โวลท์

โครงสร้างทาง HARD WARE

1) MEMORY : หน่วยความจำ

RAM 8 K

2) 8255 INPUT/OUTPUT

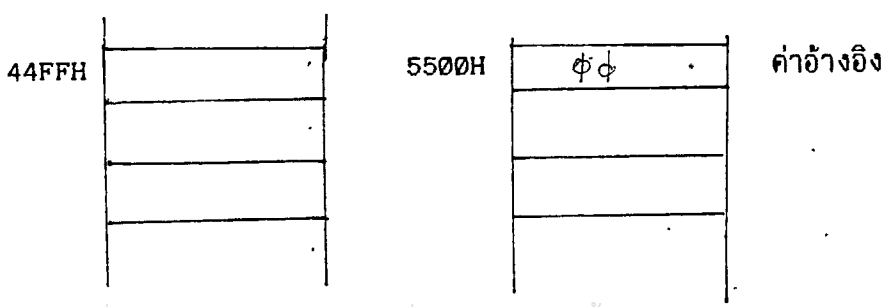
8251 INPUT/OUTPUT Series Communication โดย RS - 232

3) CPU ใช้ Z - 80 Single board

4) 74LS138 เป็นตัว Decode

5) - MC1488 เป็นตัวส่งข้อมูลจากสัญญาณ TTL เป็นสัญญาณ RS - 232

- MC1489 เป็นตัวรับข้อมูลจากสัญญาณ RS - 232 เป็นสัญญาณ TTL



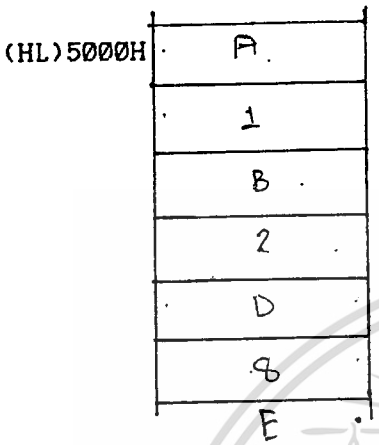
บอกโดยรีจิสเตอร์ IX

ทำหน้าที่ : เก็บข้อมูลที่ได้จากการโหลด

: เก็บข้อมูลจากผลของการกระทำ

บอกโดยรีจิสเตอร์ IY

ทำหน้าที่ : คำนวณผลลัพธ์ที่ได้จากการกระทำหรือจากการโหลด



Data ที่ชี้โดยรีจิสเตอร์ HL เป็นโปรแกรม PLC ดังตัวอย่าง ดังรูปเป็นโปรแกรม

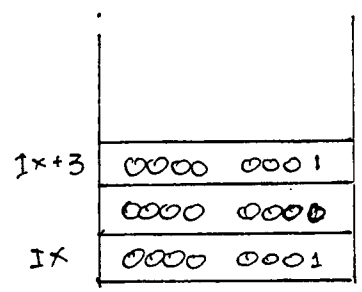
```
LD  1;
AND 2;
OUT 8;
END
```

ตำแหน่งของ Port และหมายเลข Port ต่าง ๆ จะกำหนด Decode เป็น 3 และ 20H โปรแกรมมอนิเตอร์

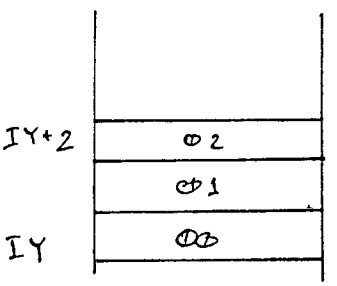
ในการเขียนโปรแกรมนั้นสามารถเขียนบนคอมพิวเตอร์ และส่งโปรแกรมที่คอมพิวเตอร์มาบน Sigle Board ได้และการส่ง compilie code ก็เช่นกันสามารถกระทำได้

โปรแกรม Compilie Code จะประกอบด้วยการทำงานของคำสั่งต่าง ๆ ของ PLC ที่รับมาจากคอมพิวเตอร์การกำหนด Code ต่าง ๆ ซึ่งแบ่งเป็น Subrutines ย่อย ๆ Monitor Subrutines

การกระทำการ Scan คำสั่งในแต่ละคำสั่งจะไม่เท่ากันในโปรแกรม Monitor ในแต่ละคำสั่งของการกระทำจะมีการเก็บค่า Status ไว้ที่ Stack เป็นชั้น ๆ ไปดังรูป

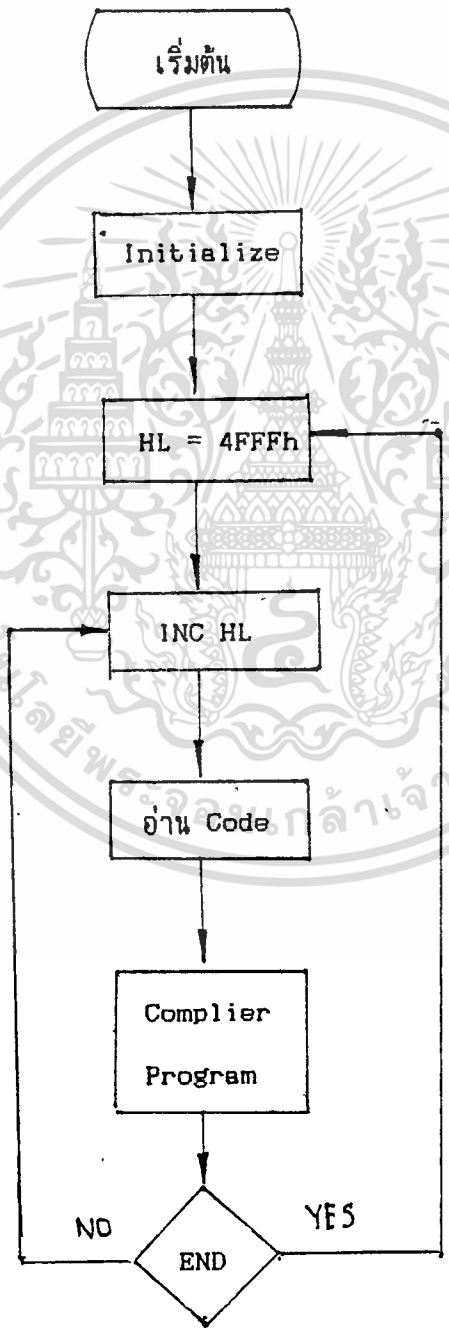


Data ที่เก็บไว้กระทำกัน



Data ที่เก็บไว้ใช้ในการกระทำย้อนกลับ

จากตัวอย่างของการกระทำคำสั่งทั้งหมดนี้ คือการนำค่าที่ Stack addr(IX + 2) มาorกับStackAddr(IX+1)ผลเก็บที่ Stack Addr(IX + 1) เลื่อน Pointer ที่ IX + 1 ถึงรู้ว่าที่ Data Addr(IX) ถึงเป็นผลในการ OUT Number bit หรือ กระทำกับคำสั่งอื่นอีก (ถ้ามี)



code7

การสแกน(Scan) คำสั่งจะนำข้อมูลจาก Memory เข้ามา Compilier จนกว่าจะพบคำสั่ง END Code จะ Reset ให้โปรแกรม Monitor ที่ Address เริ่มต้นในส่วนของ Complier นั้นจะขึ้นอยู่กับการกระทำของ Code ต่าง ๆ

โปรแกรมส่วน Compiler

ประกอบด้วย Subrutines ต่าง ๆ ดังนี้

: Call Load

เป็นการรับข้อมูลของอินพุท และถึงจะดึง Code ตัวอย่างต่อไปเข้ามาเป็นการบอกตำแหน่งของบิทที่จะนำเข้าไปเก็บไว้ใน Acccmulator บิที่ (เป็นบิท 1 ของ PLC) แล้วนำไปเก็บไว้ที่ Stack ของ PLC ที่ชี้ด้วย Regs IX

: Call And ,OR , XOR

กระทำ And ,OR ,XOR ของ Stack เดิม กับ Stack ใหม่ที่ได้จากการ Load bit เข้ามากระทำ

: Call Out

กระทำ And ,OR ของข้อมูลใน IX ที่เก็บไว้ทั้งหมดโดยการ And,OR จะกำหนดในรีจิสเตอร์ IX และส่งผลที่ได้ออก Output Port

: Call Subroutine : NOT

ประกอบด้วย Subrutine ต่าง ๆ ที่กระทำการ Inverse bit กับค่าใน Stack ก่อนหน้านั้น เช่น LDNOT ,ORNOT,ANDNOT,OUTNOT

: Call Subroutine การ AND ,OR ของส่วนย่อย ๆ แต่ละเส้นของการกระทำ

บทที่ 5

แผนผังโปรแกรม

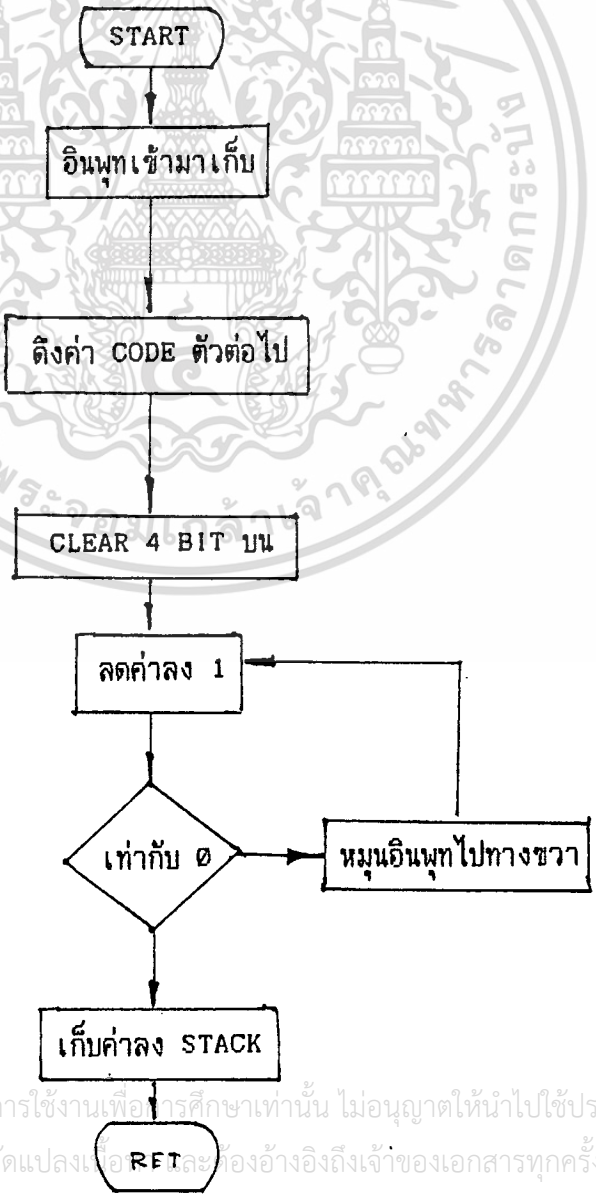
ในส่วนของโปรแกรมนี้จะแบ่งได้เป็น 2 ส่วนคือ

- โปรแกรมที่รันบน Single board เขียนเป็นภาษา แอสแซมบลี้
- โปรแกรมที่รันบน PC/XT เขียนเป็นภาษา ปาสคาล

5.1 โปรแกรมที่รันบน Single board

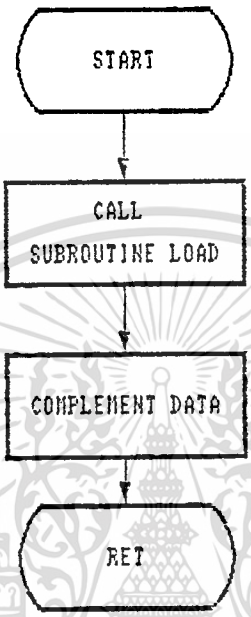
5.1.1. โปรแกรมย่อย LD Numberbit;

โปรแกรมย่อยนี้ได้ใช้สำหรับนำค่าสภาวะของอินพุท จากภายนอกเข้ามาเพื่อกระทำการใด ๆ ตามโปรแกรมควบคุม ค่าสภาวะอินพุทนำเข้ามานี้เป็นค่าในขณะที่โปรแกรมย่อยกำลังกระทำการประมวลอยู่ ค่าสภาวะอินพุทจะถูกเก็บไว้ในหน่วยความจำที่ IX ชี้อยู่



5.1.2. โปรแกรมย่อย LD NOT Numberbit;

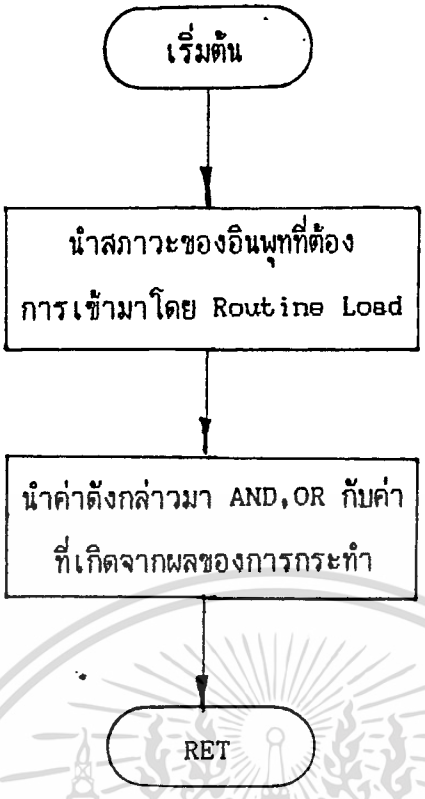
โปรแกรมย่อยนี้ใช้สำหรับอินพุทในสภาวะปกติไม่ทำงาน เมื่อรับสภาวะอินพุทเข้ามาแล้วก็กลับสภาวะอินพุทให้อยู่ในสภาวะตรงกันข้าม คือถ้าอินพุทไม่ทำงานเมื่อกลับสภาวะแล้วก็จะทำงาน คือ มีอินพุทนั่นเอง



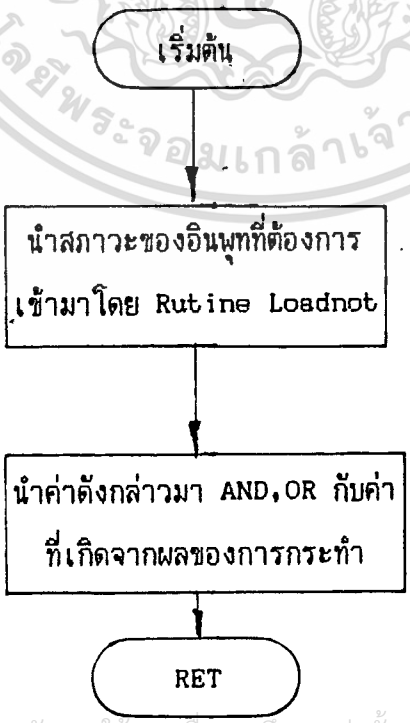
SUBROUTINE LOADNOT N;

5.1.3. โปรแกรมย่อย AND Numberbit; หรือ OR Numberbit;

โปรแกรมย่อยนี้ใช้ในการนำค่าสภาวะของอินพุทมากระทำฟังก์ชั่นทางลอจิก AND หรือ OR กับสภาวะเก็บไว้ใน Stack ตามเดิมเพื่อใช้ในกระทำฟังก์ชั่นต่าง ๆ กับคำสั่งถัดไป



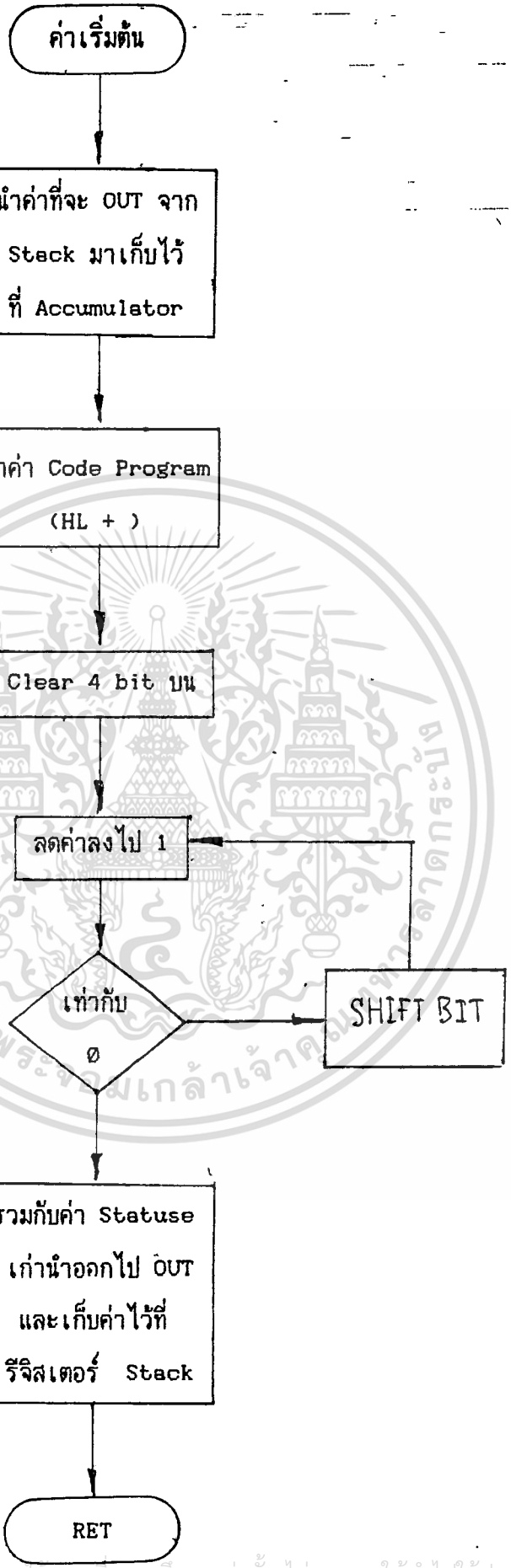
5.1.4. โปรแกรมย่อย AND NOT Numberbit และ ORNOT Numberbit; เช่นเดียวกัน โปรแกรมย่อยนี้ใช้สำหรับจำค่าสภาวะที่ตรงข้ามกันกับความจริงเข้ามากระทำฟังก์ชั่นลอจิก AND หรือ OR กับสภาวะที่เก็บไว้ใน Stack ผลลัพธ์ที่ได้จากการกระทำดังกล่าวจะเก็บไว้ในริจิสเตอร์ตามเดิม



5.1.5. โปรแกรมย่อย XOR Numberbit;

ลักษณะคล้ายโปรแกรมย่อย AND , OR ดังรูป

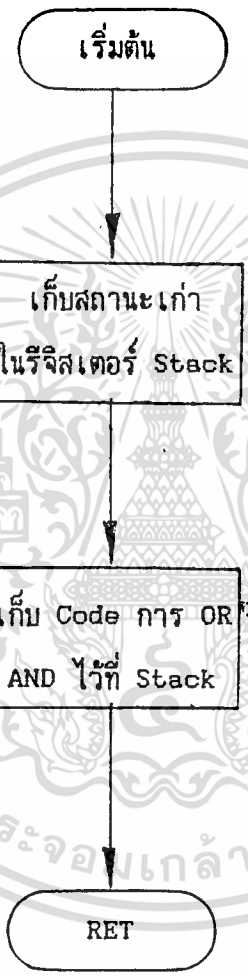
5.1.6. โปรแกรมย่อย OUT Numberbit;



ก่อนที่จะม่การ OUT ตัวโปรแกรม Complier จะมีการกระทำค่าใน Address Stack ก่อนซึ่งลักษณะการกระทำจะบอกได้จาก Stack ที่บอกโดยรีจิสเตอร์ IY

5.1.7. โปรแกรมย่อย ... (;....) และ ....OR(;...)

การกระทำของ Subrutine นี้จะเก็บสถานะเก่าและที่จะ Load เข้าไปใหม่มากระทำ AND , OR กันในขณะที่พบ Code ปิด



5.2 โปรแกรมที่รันบน PC/XT เขียนเป็นภาษา ปาสคาล

เป็นส่วนที่ควบคุมความผิดพลาดต่าง ๆ ส่วน คอนโทรลและส่วนแปลงรหัสโปรแกรมดังโปรแกรมซึ่งสามารถแบ่งการทำงานได้ดังนี้

5.2.1 ส่วนควบคุมความผิดพลาด

จะมีการตรวจสอบรูปแบบการเขียนโปรแกรมต่างพร้อมทั้งบอกบรรทัดที่ผิดพลาดด้วย

การเขียนโปรแกรมนั้นจะต้องมีเครื่องหมายการจบบรรทัดเสมอและสามารถเขียนรายละเอียดได้ภายหลังเขียนคำสั่งจบ (;) แล้ว

ตัวอย่างเช่น

```
LD 1;  LOAD BIT 1 TO ACCUMULATOR , STACK
AND (;
LD 2; INCREASE STACK TO STORE
OR 3);
OUT 4;
END
```

5.2.2 ส่วน Control นั้น คือส่วนที่กดคีย์บน Keyboard ควบคุมการรับหรือการป้อนโปรแกรมใหม่ให้คีย์เป็น R , S ได้โดย R ; จะใช้เมื่อต้องการรันโปรแกรม และ S ; ใช้สำหรับต้องการป้อนโปรแกรมใหม่

5.2.3 ส่วนแปลงรหัสเป็นโค๊ดเพื่อที่จะส่ง Code ไปยัง Sigle Board Z - 80 ทำการ Complier ต่อไป โค๊ดที่ส่งไปจะถูกต้องเสมอ ซึ่งส่วนนี้ไม่ต้องมีกาตรวจสอบบนโปรแกรม Complier เลย ตัวอย่างโปรแกรม เช่น

ตัวอย่างโปรแกรม

LD 1;	ตัวโค๊ดที่ Memory 5 H คือ	41	31		
ORC ;		42	32		
LD 7;		4B			
AND 8);		42	38	29	
ANDNOT 3;		47	33		
OUT 2;		44	32		
END	5 E	45			

5.2.4 ส่วนของการจัดรูปแบบโปรแกรมจัดการ (บน Sigle Board) พร้อมทั้งส่งให้ Sigle Board จัดการ

ในส่วนนี้มีชื่อว่า "CONVERTHEXFILE" ซึ่งจะแปลงโปรแกรมการจัดการซึ่งอยู่ในรูปของ HEXFILE (ที่ได้จากการ Complier ด้วย C = 32 Complier) เป็นข้อมูลแบบ Byte พร้อมที่จะส่งผ่าน RS = 232 ให้ Sigle Board เช่น

```
:104000003ECED3213E27D3213E82D3331814DB2169
:10401000CB4F28FADB20C947DB21CB4728FA78D3DE
:1040200020C9C32C403E55CD174018F9210050CD72
:10403000E40FE452804772318F57776CD0E40CD47
:04404000174018F815
:00000001FF
```

รูป HEXFILE ของโปรแกรมการจัดการ

บทที่ 6

สรุปผลและวิจารณ์

จากข้อคิดต่างๆ และแนวทางที่กำหนดไว้จึงได้ค้นคว้าและออกแบบส่วนประกอบต่างๆ ทั้งทางด้าน HARDWARE และ SOFTWARE เพื่อที่จะประกอบกันให้ทำงานร่วมกัน ซึ่งแบ่งเป็น 2 ส่วนดังนี้คือ

1. การพัฒนาทางด้าน HARDWARE

2. การพัฒนาทางด้าน SOFTWARE

สรุปการพัฒนาทางด้าน HARDWARE

ในการพัฒนาทางด้าน HARDWARE จะใช้ CHIP SUPPORT ที่จำเป็นและมีแต่ละส่วน

1. SINGLE BOARD Z80 ของภาควิชา CONTROL

2. 8 DIGITAL OUTPUT PORT

3. 8 DIGITAL INPUT PORT

4. SERIAL DATA COMMUNICATION (ขณะนี้เป็นส่วนเดียวที่ยังมีการผิดพลาดในการติดต่อ)

สรุปผลการพัฒนาทางด้าน SOFTWARE

ในการพัฒนาทางด้าน SOFTWARE จะพิจารณาได้จาก FLOW CHART ที่ให้ไว้ เราสามารถใช้คำสั่งต่างๆ และกระทำการต่างๆ ให้รองรับ HARDWARE ซึ่งส่วนใหญ่จะเขียนเป็นโปรแกรมย่อยเอาไว้

เราจะแบ่งส่วน SOFTWARE ออกเป็น 2 ส่วน

1. SOFTWARE ทางด้านเครื่อง COMPUTER ในโครงงานนี้ใช้ภาษา PASCAL ในการเขียนโปรแกรม โดยทำหน้าที่แปลงภาษา PLC (ที่กำหนดไว้) เป็น OPCODE PLC และส่ง FILE ของ OPCODE นั้น พร้อมทั้งตรวจสอบรูปแบบการเขียนโปรแกรม และมีรหัสควบคุมสำหรับ SINGLE BOARD

2. ส่วนของโปรแกรมที่รันโดย Z-80 CPU เป็นส่วนการจัดการระบบซึ่งจะมีส่วน Initialize ส่วนแปล OPCODE PLC และส่วนโปรแกรมย่อยของคำสั่งแต่ละ OPCODE ซึ่งสามารถรองรับ HARDWARE ที่ได้สร้างขึ้น

วิจารณ์

ในส่วนการออกแบบ

ในโครงงานครั้งนี้ ช่วงต้นโครงงาน ได้ใช้ SINGLE BOARD 8085 CPU แต่มีปัหาทางด้าน BOARD จึงจำลอง PLC ขึ้นมาโดยจะใช้ HARDWARE และ CHIF SUPPORT สำหรับ Z-80 และใช้ CPU Z-80 ในส่วน HARDWARE จะต้องสามารถให้ OUTPUT และสามารถรับ INPUT หรือกระทำขบวนการต่าง ๆ ได้ เช่นเดียวกับ PLC ในลักษณะที่หน่วยประมวลผลกลางจะทำหน้าที่ควบคุมการทำงานทั้งหมด ฉะนั้นต้องมี PROGRAM การจัดการให้ CPU ซึ่งจะเขียนเป็นโปรแกรมการจัดและเราจะสร้าง OPCODE ของ PLC ขึ้นมา โดยการจัดการจะรองรับ OPCODE PLC เหล่านี้ และเขียน PROGRAM PLC เป็น OPCODE PLC ไว้ในตำแหน่งส่วน PROGRAM PLC

จากนี้ SINGLE BOARD ก็จะสามารถ RUN ให้เป็น PLC ได้ แต่มีส่วนเพิ่มเติมคือ เราจะไม่ต้อง KEY PROGRAM PLC นี้ แต่จะส่งส่วน PROGRAM นี้มาจาก COMPUTER ซึ่งเป็นการควบคุมจาก COMPUTER

ในส่วนของ HARDWARE

ได้มีการต่อ PORT ออกจาก SIGLE BOARD Z-80 โดยใช้ 8255 เป็น I/O PORT 8 BIT , MEMORY ใช้บน SIGLE BOARD ขนาด RAM 8K สำรองไว้อีก 8K มี 8251 เป็น HARDWARE ติดต่อกับ MULTI I/O บนคอมพิวเตอร์ ใช้สำหรับส่งผ่านข้อมูลเข้ามาเก็บไว้ที่ MEMORY โปรแกรม และส่งส่วนของ I/O PORT สามารถต่อเพิ่มได้อีก วงจรส่วน DECODE สามารถต่อขยายภายนอกได้แต่ PORT ที่ DECODE ให้บน PORT ใช้ DECODE เป็น 30H เป็น PORT OUTPUT , 31H เป็นชื่อ PORT INPUT ส่วน PORT บน 8255 ใช้หมายเลข 20H และ 21H ส่วนวงจรคล๊อกจะสร้าง 19.2 KHz กำหนดการรับส่งข้อมูล 1200 BAUD RATE

ส่วนของ SOFTWARE

ได้แบ่งเป็น 2 ส่วน คือ ส่วนบนคอมพิวเตอร์ และส่วน COMPLIER ภาษา ASSEMBLY ภาษาที่เขียนจะอยู่ในรูปที่เข้าใจง่าย จำนวนคำสั่งจะมีน้อยในภาษา PLC จริงๆ แล้วต้องมีคำสั่งเกี่ยวกับการใช้ TIMER คำสั่งในการกระโดด ซึ่งเป็นส่วนของการใช้ INTERRUPT และลักษณะทางด้าน HARDWARE จะมี CTC (TIMER ,

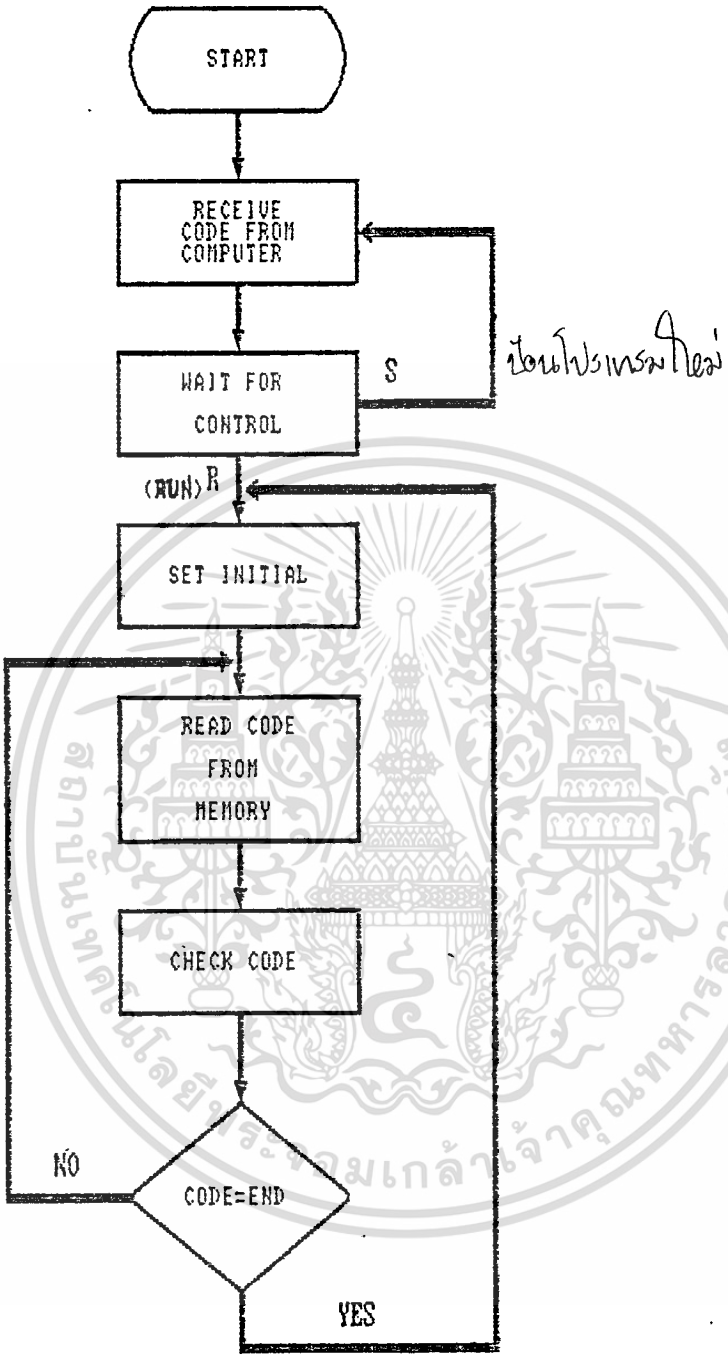
COUNTER) แต่ขณะนี้ได้สิ้นสุดเวลาของโครงงานจึงไม่สามารถพัฒนาต่อ

ส่วนโปรแกรมบนคอมพิวเตอร์นั้นจะประกอบด้วย SUBRUTINE ย่อยคือส่งไฟล์โปรแกรม ตรวจรูปแบบการเขียนส่วนควบคุมในการกดรันและป้อนโปรแกรมใหม่ ส่วนที่ส่งไฟล์ภาษา ASSEMBLY ซึ่งไม่ต้องการอาศัยการกดคีย์ก็สามารถส่งเข้าสู MEMORY ได้เลย

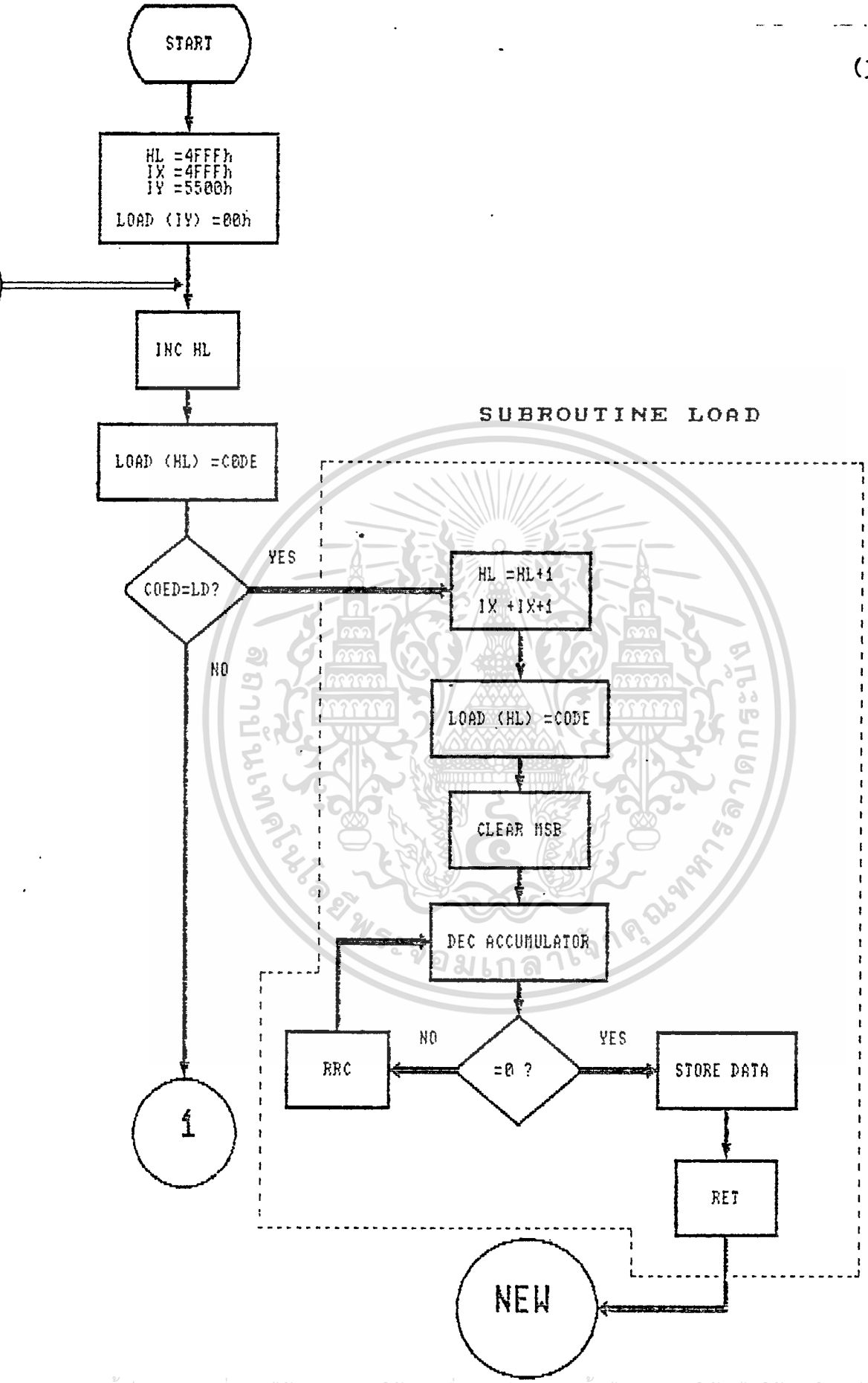
สำหรับผู้ที่พัฒนาต่อ

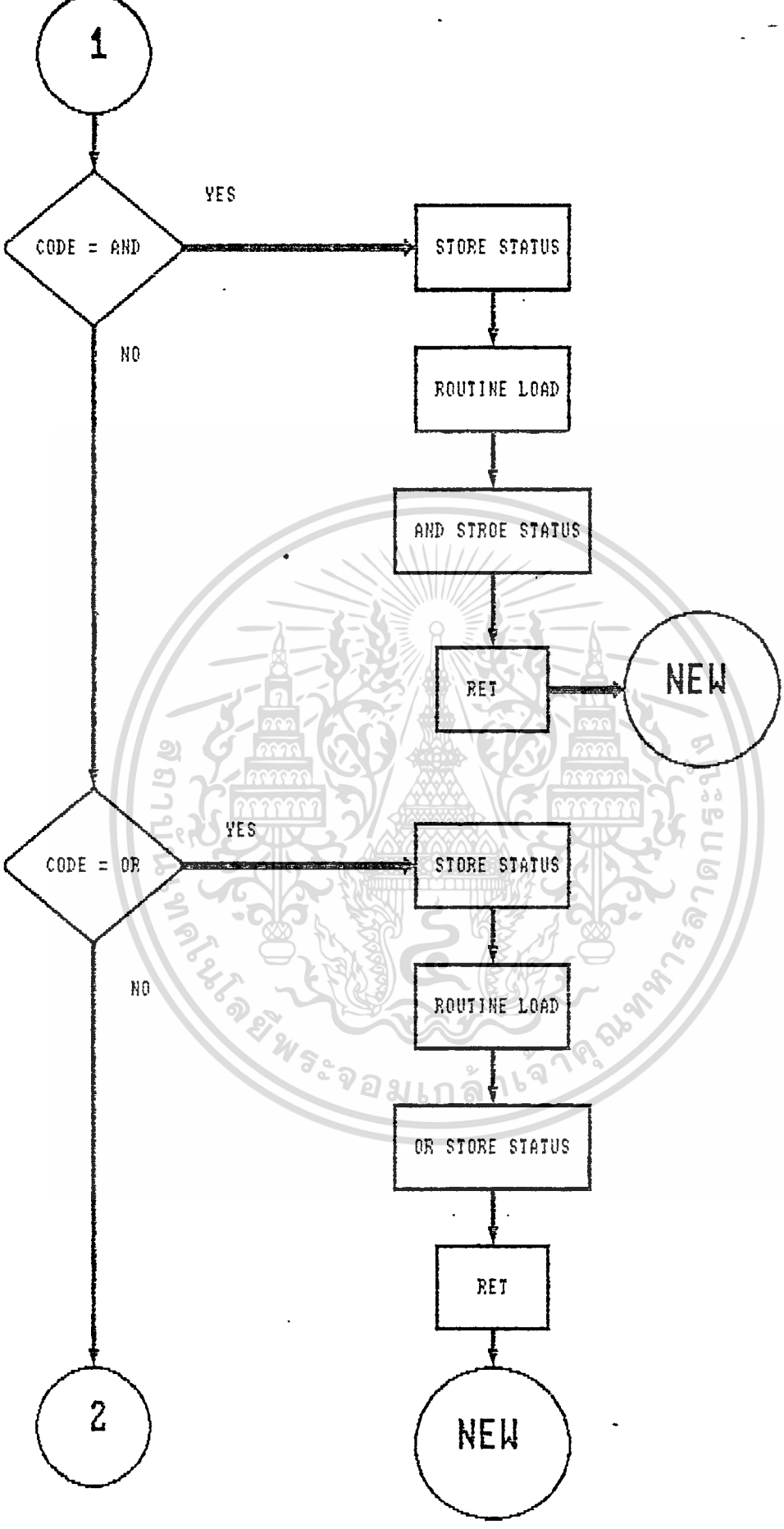
ซึ่งผู้ใช้สามารถนำปริญญานิพนธ์นี้ไปศึกษาต่อได้โดยสำหรับผู้ที่พัฒนาต่อ เช่น การแปลงภาษาสูงเป็นASSEMBLYของ Z-80 สามารถเขียน SUBROUTINE คำสั่งของภาษาสูงขึ้นมาในรูปของภาษา ASSEMBLY และกำหนด CODE คำสั่งของคำสั่งนั้นขึ้นมา จากนั้นก็ส่งโปรแกรมทั้งหมดให้ SIGLE BOARD ได้เช่นเดียวกัน การส่งโปรแกรม PLC ในปริญญานิพนธ์ฉบับนี้ ส่วนการกำหนด CODE คำสั่ง เองใหม่นี้จะต้องเข้าไปแก้ไขในส่วน PROCESDURE CHANGDATA โดยเปลี่ยนตาราง OPCODE และชื่อของคำสั่งต่าง ๆ แต่ขณะนี้ได้สิ้นสุดเวลาของโครงงานจึงไม่สามารถพัฒนาได้ต่อ

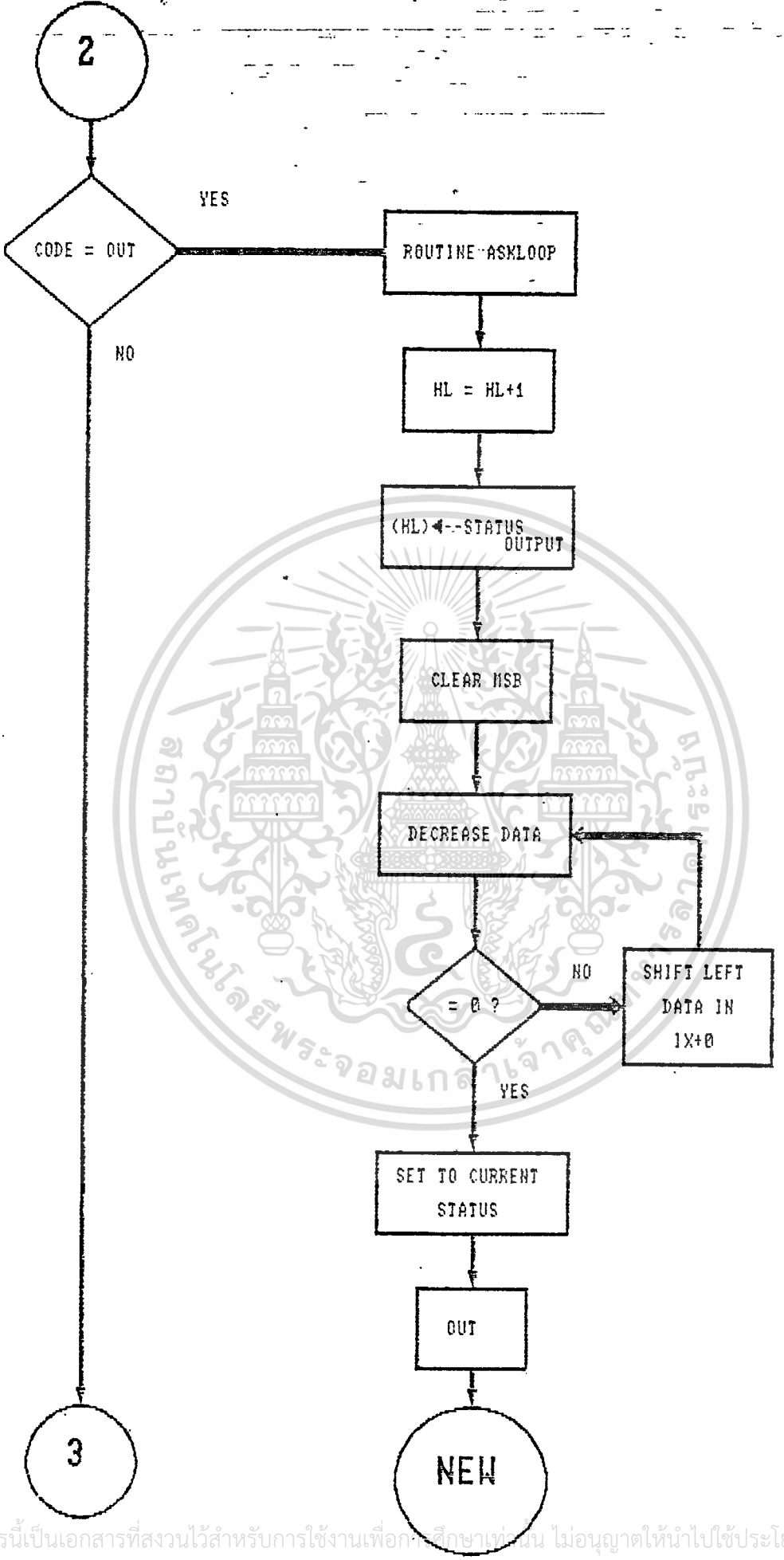
ภาคผนวก ก FLOW CHART การทำงานของโปรแกรม

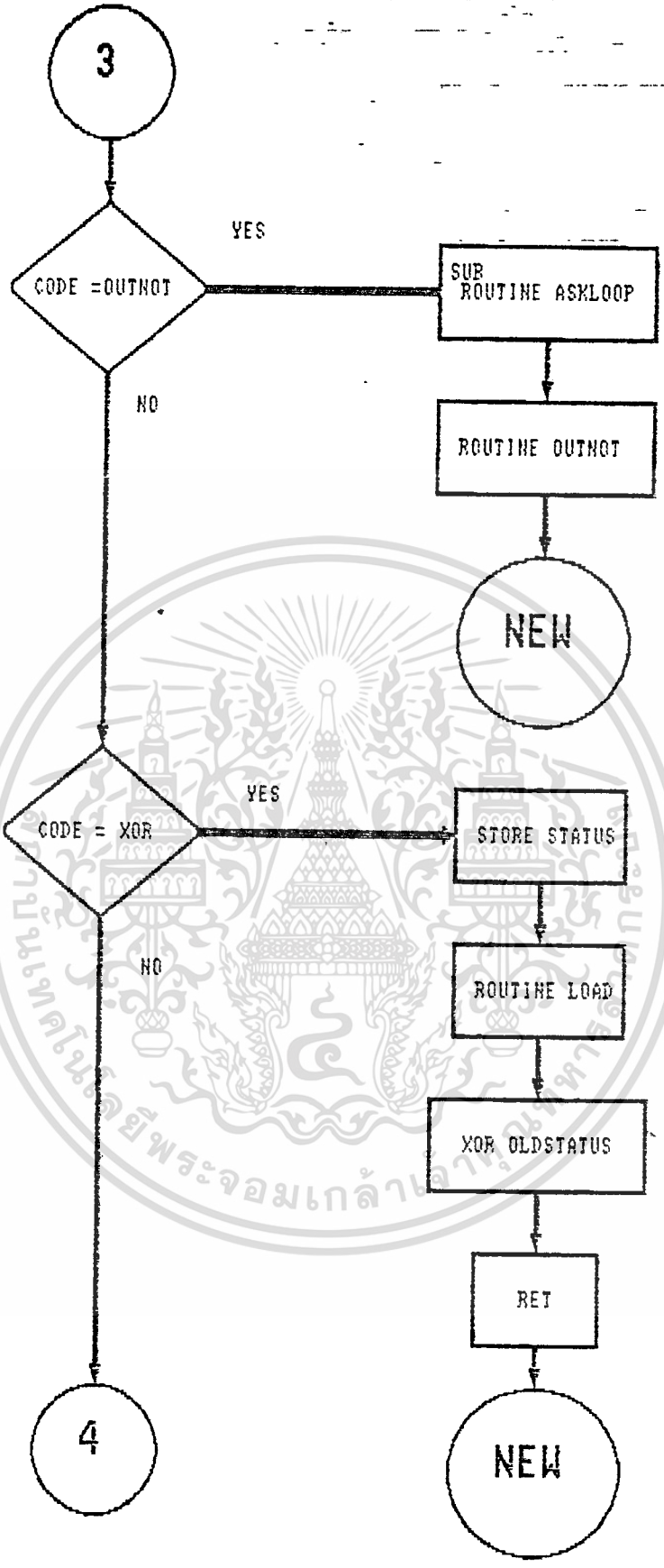


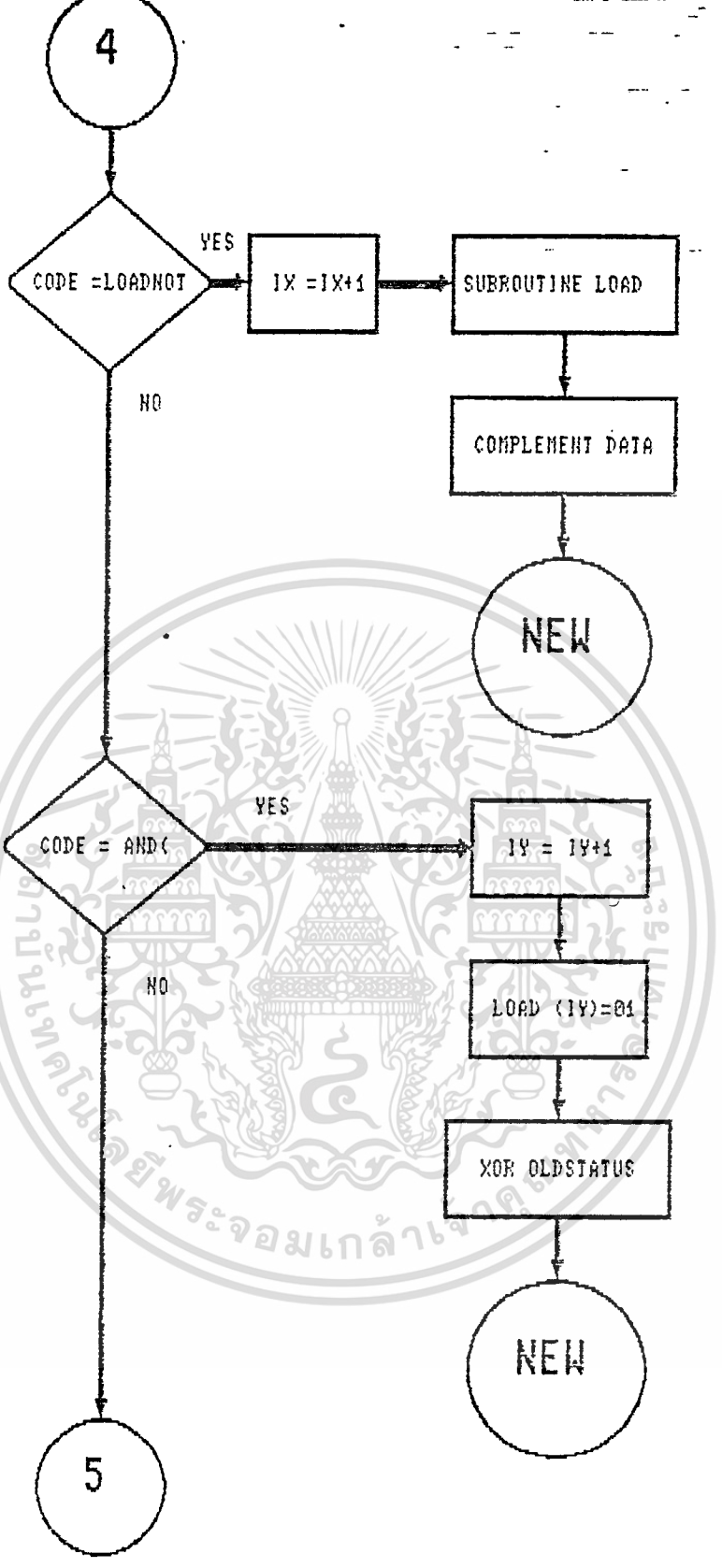
MAIN FLOW CHART

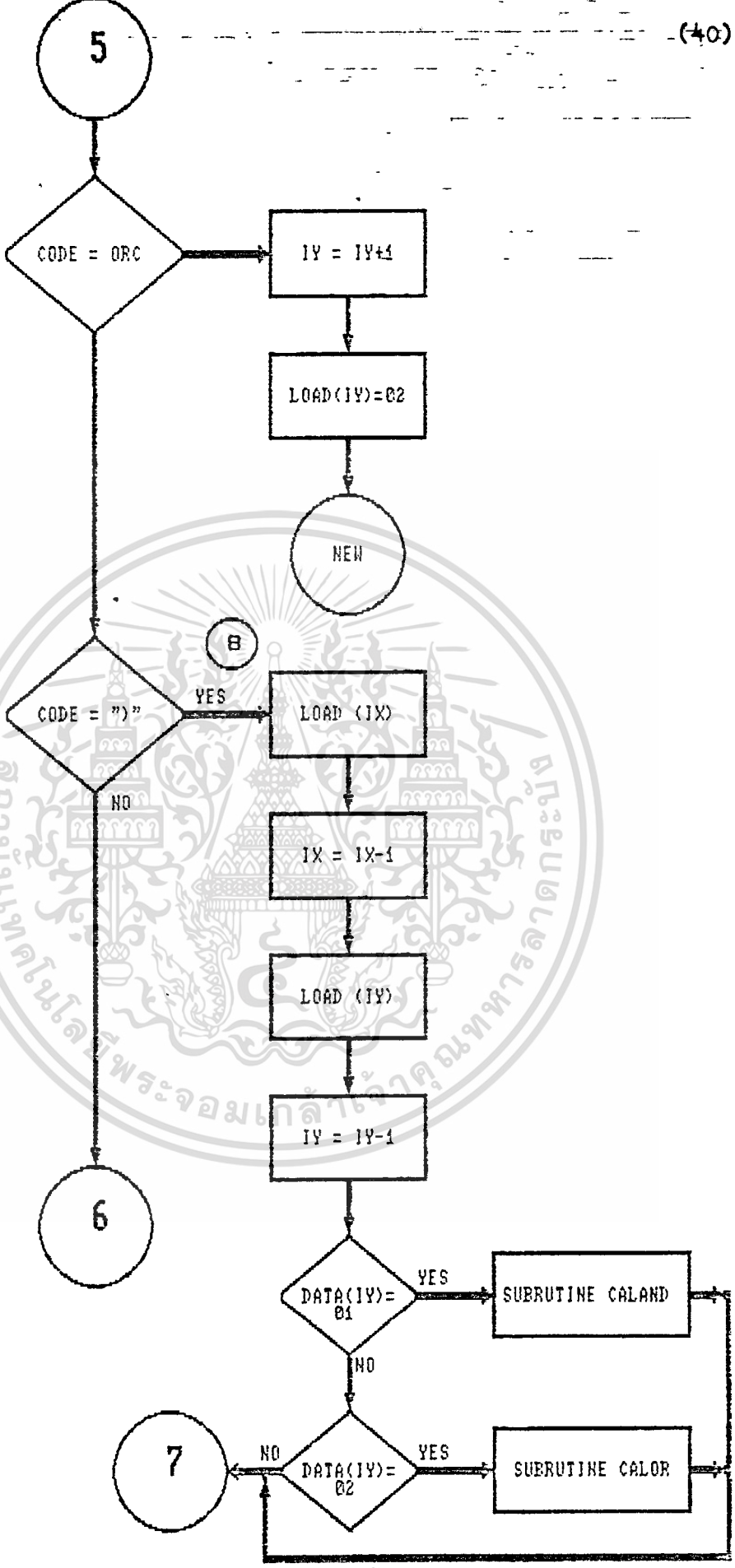


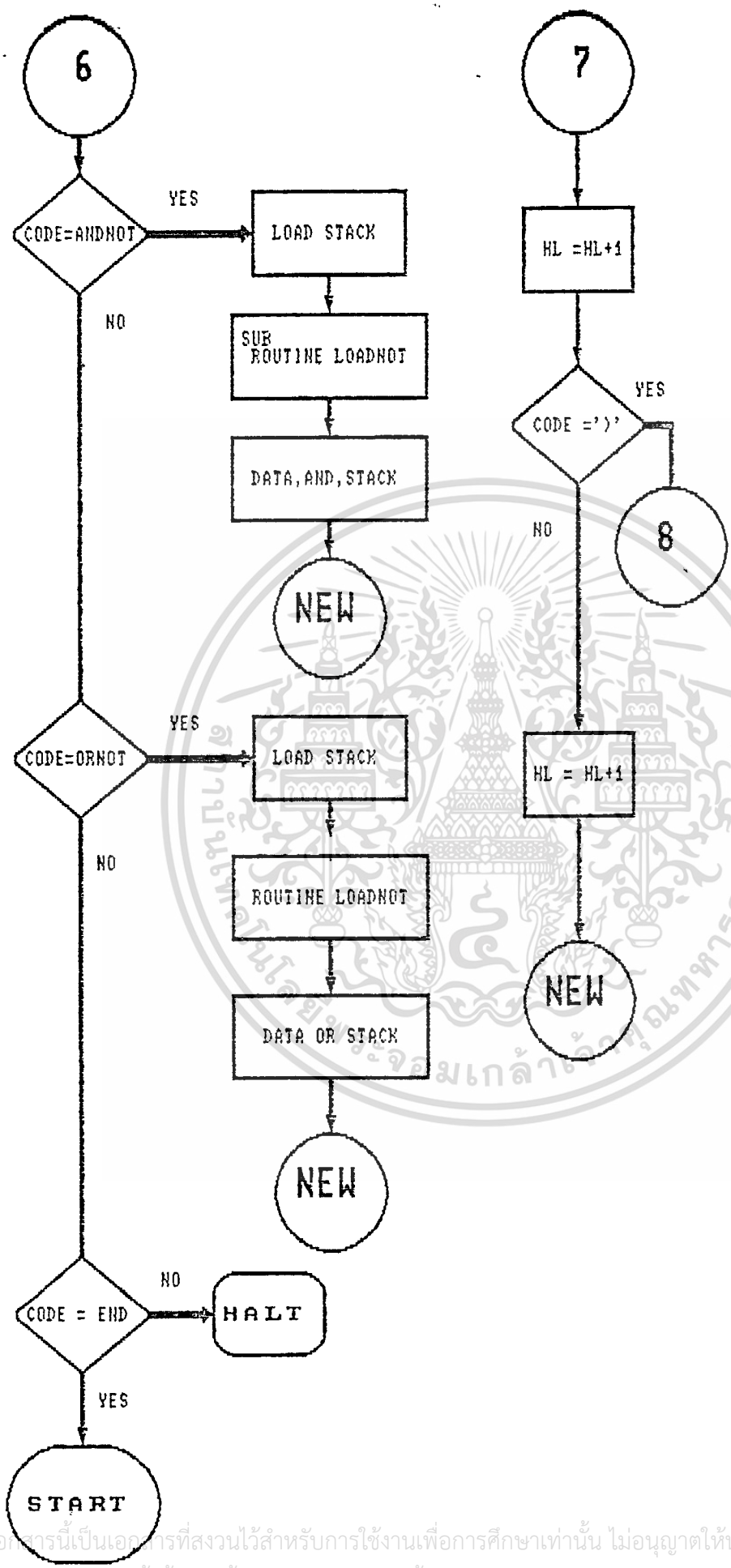




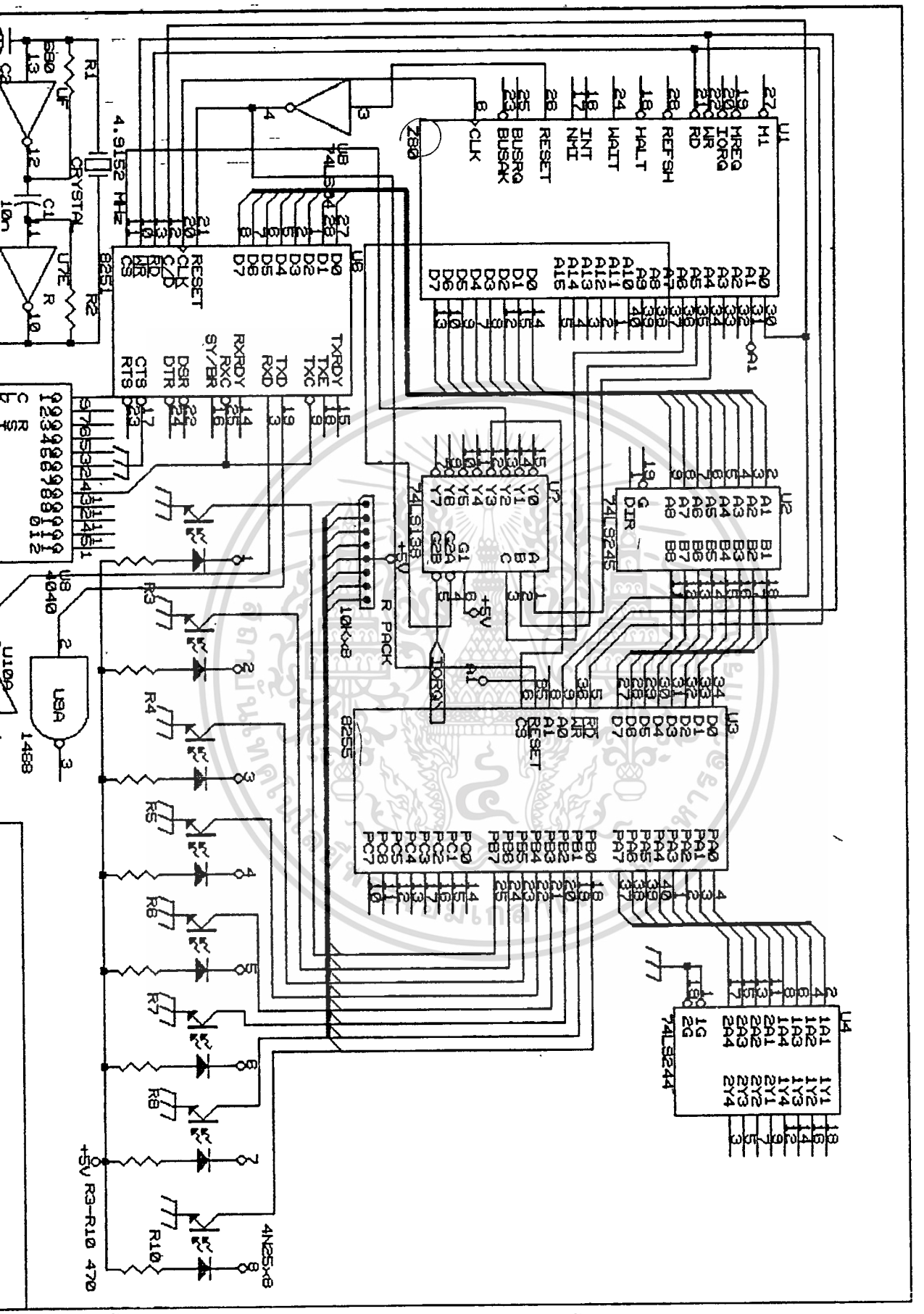




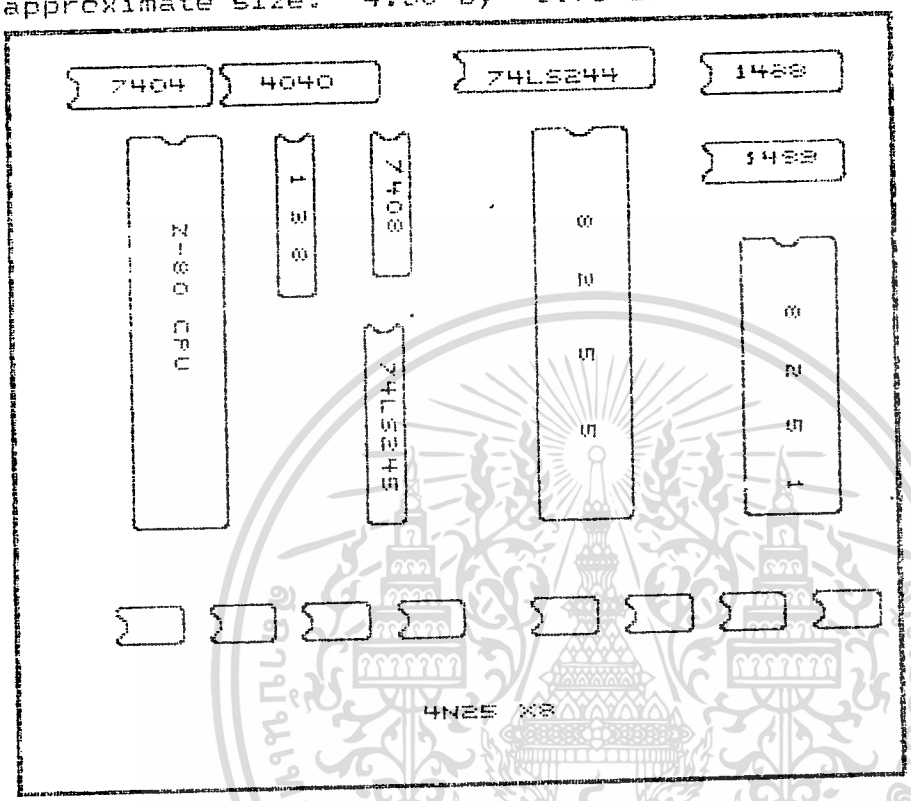




ภาคผนวก ข วงจร HARDWARE ของ BOARD



1X checkplot 27 Mar 1990 00:01:48
b:circuit\pad3
v1.2 r3 holes: 285 silkscreen
approximate size: 4.50 by 3.75 inches



แผนผัง ตำแหน่งไอซี

ภาคผนวก ค โปรแกรมการทำงาน

```
          CPU "z80.TBL"
          HOF "INT8"

SERDTA:   EQU   20H             ; SERIAL DATA    PORT
SERCOM:   EQU   21H             ; SERIAL CONTROL PORT
PORT_A:   EQU   30H             ; OUTPUT  PORT
PORT_B:   EQU   31H             ; INPUT   PORT
PORT_C:   EQU   32H             ; INPUT   PORT
PORTCON:  EQU   33H             ; 8255  CONTROL PORT

org  4000h
COMINI: LD      A,0CEH          ;  {                    }
        OUT     (SERCOM),A      ;  { 8251 INITIALIZE }
        LD      A,27H           ;  {                    }
        OUT     (SERCOM),A      ;  {                    }
        LD      A,82H           ;  { 8255 INITIALIZE }
        OUT     (PORTCON),A     ;  A & C = OUTPUT , B = INPUT
        JR      START

; INPUT A CHARACTER INTO THE ACCUMULATOR FROM THE 8251

COMIN:  IN      A,(SERCOM)      ;WAIT FOR CHARACTER TO BE SENT
        BIT     1,A
        JR      Z,COMIN         ;COMIN
        IN      A,(SERDTA)
        RET

; OUTPUT A CHARACTER IN THE ACCUMULATOR

COMOUT: LD      b,A
comout0:IN      A,(SERCOM)
        BIT     0,A
        JR      Z,COMOUT0       ;COMOUT0
        LD      A,b
        OUT     (SERDTA),A
        RET

START:  JP INDATA
OUTDATA:LD A,55H
        CALL COMOUT
        JR OUTDATA
INDATA: LD HL,5000H
insrl:  CALL  COMIN
        CP 45H                  ; ASCII 'END'
        JR Z,FINAL
        LD (HL),A
        INC HL
        JR INSRL
FINAL:  LD (HL),A
        HALT

INOUT:  CALL COMIN
        CALL COMOUT
        JR INOUT

  EN
```

```
                        ; ##############################################
                        ; ##############################################
0000                            CPU"Z80.TBL"
0000                            HOF"INT8"
4000                            ORG 4000H

                        ;----------------------------------------------

                        ; REGS A   = ACCUMULATOR BIT 0
                        ; REGS B   = MARK SHIFT REGISTER (0FEH)
                        ; REGS C   = ACCUMUTOR STACK
                        ; REGS D   = ZERO POSITION LOAD SHIFT REGISTE
                        ; REGS E   = STATUS OF DIGITAL OUTPUT
                        ; REGS HL  = PLC PROGRAM POINTER
                        ; REGS IX  = STATUS LINE STACK PNTR ADDR 4500
                        ; REGS IY  = MAKE LOGIC STACK PNTR ADDR 5500H
                        ; LOAD PROGRAM PLC START ADDR 5000H
                        ; STACK PLC AT REGS IX  (STATUS ACC BIT)

                        ; ##############################################
                        ; ##############################################

4000 3E82       START:  LD A,082H       ;INITIAL PORT ON HARDWARD(8255
4002 D333               OUT (033H),A
4004 3ECE               LD A,0CEH       ; MODE INITIAL PORT ON 8251
4006 D321               OUT (021H),A
4008 3E27               LD A,027H       ; COMMAND INITIAL PORT ON 8251
400A D321               OUT (021H),A
400C 1E00               LD E,00H        ;START DATA OUT PORT  00H
400E 7B                 LD A,E
400F D330               OUT (030H),A    ;OUTPUT START DATA
4011 210050             LD HL,05000H    ;BASE PNTR STORE ADDR PROG PLC
4014 DD21FF44           LD IX,044FFH    ;ORIGIN BASE ADDRESS
                                          OF STACK PNTR ON PLC
4018 FD210055           LD IY,05500H    ;BASE STACK OF MONITER PROG PL
401C FD360000           LD (IY+0),00H;DATA BASE OF MON PROG PLC
4020 DB21       CALL:   IN A,(021H)     ;(LOOP CALL FOR READ
                                        PROG  MEM AT 5000H)
4022 FE02               CP 02H
4024 20FA               JR NZ,CALL
4026 DB20               IN A,(020H)
4028 77                 LD (HL),A
4029 FE45               CP  045H        ;IF END OF FILE JUMP CONTROL
402B 2803               JR Z,CONTROL
402D 23                 INC HL
402E 18F0               JR CALL
4030 DB21       CONTROL: IN A,(021H)
4032 FE02               CP 02H
4034 20FA               JR NZ,CONTROL
4036 DB20               IN A,(020H)
4038 FE52               CP 052H         ;(R)UN PROGRAM
403A 2806               JR Z,RUN1
403C FE53               CP 053H
403E 28C0               JR Z,START      ;(S)TORT NEW DATA
4040 18EE               JR CONTROL

                ;PROCEDURE RUN MAIN PROGRAM
4042 21FF4F RUN1:       LD HL,4FFFH     ;START RUN PROGRAM
4045 23     RETR:       INC HL
4046 7E                 LD A,(HL)
4047 FE41               CP 041H         ;LD  COMMENT
4049 2007               JR NZ,LOOP1
404B DD23               INC IX
404D CDAA41             CALL LOADR
4050 18F3               JR RETR
4052 FE42   LOOP1:      CP 042H         ;AND COMMENT
4054 2005               JR NZ,LOOP2
4056 CD2C41             CALL ANDR1
```

```
4059 18EA              JR RETR
405B FE43   LOOP2:     CP 043H          ;OR  COMMENT
405D 2005              JR NZ,LOOP3
405F CD3B41            CALL ORR1
4062 18E1              JR RETR
4064 FE44   LOOP3:     CP 044H          ;OUT COMMENT
4066 200C              JR NZ,LOOP4
4068 CD1541            CALL ASKLOOP
406B CD4A41            CALL OUTR1
406E DD21FF44          LD IX,044FFH     ;OR , DEC IX  CLR DATA
4072 18D1              JR RETR
4074 FE4C   LOOP4:     CP 04CH          ;OUTNOT COMMENT  INVERSE RUN
4076 200C              JR NZ,LOOP5
4078 CD1541            CALL ASKLOOP
407B CD6641            CALL OUTNOT1
407E DD21FF44          LD IX,044FFH     ;OR DEC IX CLR DATA OUT
4082 18C1              JR RETR
4084 FE46   LOOP5:     CP 046H          ;LDNOT  COMMENT
4086 2007              JR NZ,LOOP12
4088 DD23              INC IX
408A CDA041            CALL LDNOTR
408D 18B6              JR RETR
408F FE47   LOOP12:    CP 047H          ;ANDNOT COMMENT
4091 2005              JR NZ,LOOP6
4093 CD8241            CALL ANDNOT1
4096 18AD              JR RETR
4098 FE48   LOOP6:     CP 048H          ;ORNOT  COMMENT
409A 2005              JR NZ,LOOP7
409C CD9141            CALL ORNOT1
409F 18A4              JR RETR
40A1 FE49   LOOP7:     CP 049H          ;XOR    COMMENT
40A3 2005              JR NZ,LOOP8
40A5 CD7341            CALL XOR1
40A8 189B              JR RETR
40AA FE4A   LOOP8:     CP 04AH          ;'AND ('   COMMENT
40AC 2009              JR NZ,LOOP9
40AE 3E01              LD A,01H         ;CODE FOR MARK IS 01H =AND
40B0 FD23              INC IY
40B2 FD7700            LD (IY+0),A      ;MARK MEMORY FOR 'AND'
40B5 188E              JR RETR
40B7 FE4B   LOOP9:     CP 04BH          ;'OR ('    COMMENT
40B9 2009              JR NZ,LOOP10
40BB 3E02              LD A,02H         ;CODE FOR MARK IS 02H =OR
40BD FD23              INC IY
40BF FD7700            LD (IY+0),A      ;MARK MEMORY FOR 'OR'
40C2 1881              JR RETR
40C4 FE29   LOOP10:    CP 029H          ;')' CLOSE LOOP  CALCULATE
40C6 200D              JR NZ,LOOP11
40C8 CDDD40  RETR1:    CALL DOT
40CB 23                INC HL
40CC 7E                LD A,(HL)
40CD FE29              CP 029H
40CF 28F7              JR Z,RETR1
40D1 2B                DEC HL
40D2 C34540            JP RETR
40D5 FE45   LOOP11:    CP 045H          ;END COMMENT
40D7 C24540            JP NZ,RETR       ;CHECK ERROR PROGRAM PLC
40DA C34240            JP RUN1          ;END FILE OF MAIN PROGRAM
                  ;*START PROCEDURE LOOP*
40DD FD7E00     DOT:   LD A,(IY+0)      ;LOOP FOR LINE ASK
40E0 FE01              CP 01H
40E2 2005              JR NZ,DOT1
40E4 CDF540            CALL CALAND
40E7 180A              JR DOT2
40E9 FE02       DOT1:  CP 02H
40EB 2005              JR NZ,HALT3      ;NO COMMENT
40ED CD0541            CALL CALOR
40F0 1801              JR DOT2
40F2 76                HALT3:HALT
40F3 00         DOT2:  NOP
```

```
40F4 C9                  RET

40F5 DD7E00 CALAND:  LD A,(IX+0)     ;(DATA (IX)) AND
                          ; (DATA (IX-1)) STORE IN DATA(IX-1
40F8 4F                  LD C,A
40F9 DD2B                DEC IX
40FB DD7E00              LD A,(IX+0)
40FE A1                  AND C
40FF DD7700              LD (IX+0),A
4102 FD2B                DEC IY
4104 C9                  RET

4105 DD7E00  CALOR:  LD A,(IX+0) ;store(DATA(IX))OR(DATA(IX-1))
4108 4F                  LD C,A
4109 DD2B                DEC IX
410B DD7E00              LD A,(IX+0)
410E B1                  OR C
410F DD7700              LD (IX+0),A
4112 FD2B                DEC IY
4114 C9                  RET

4115 FD7E00  ASKLOOP:LD A,(IY+0)
4118 FE01                CP 01H
411A 2005                JR NZ,ASKLOOP1
411C CDF540              CALL CALAND
411F 18F4                JR ASKLOOP
4121 FE02    ASKLOOP1:CP 02H
4123 2005                JR NZ,ASKLOOP2 ;IF DATA IN ADDR(REGS IY) =0
4125 CD0541              CALL CALOR
4128 18EB                JR ASKLOOP
412A 00      ASKLOOP2:NOP
412B C9                  RET

412C DD7E00   ANDR1: LD A,(IX+0)
412F 4F                  LD C,A
4130 CDAA41              CALL LOADR
4133 DD7E00              LD A,(IX+0)
4136 A1                  AND C
4137 DD7700              LD (IX+0),A
413A C9                  RET

413B DD7E00              ORR1: LD A,(IX+0)
413E 4F                        LD C,A
413F CDAA41                    CALL LOADR
4142 DD7E00                    LD A,(IX+0)
4145 B1                        OR C
4146 DD7700                    LD (IX+0),A
4149 C9                        RET

414A 23                  OUTR1: INC HL
414B DD7E00                     LD A,(IX+0)
414E 4F                         LD C,A      ;DATA TO OUT EX'(BIT 1)'
414F 06FE                       LD B,0FEH   ;11111110
4151 7E                         LD A,(HL)
4152 E60F                       AND 00FH
4154 3D                  OUTR4: DEC A
4155 FE00                       CP 00H
4157 2806                       JR Z,OUTR3
4159 CB01                       RLC C
415B CB00                       RLC B
415D 18F5                       JR OUTR4
415F 7B                  OUTR3: LD A,E
4160 A0                         AND B       ;CLR BIT OUT,OTHER IS CONS
4161 B1                         OR C        ;OR BIT SHIFT
4162 5F                         LD E,A      ;STORE NEW STATE
4163 D330                       OUT (30H),A
4165 C9                         RET

4166 DD7E00          OUTNOT1:  LD A,(IX+0)
4169 2F                        CPL
```

```
416A E601                       AND 01
416C DD7700                     LD (IX+0),A
416F CD4A41                     CALL OUTR1
4172 C9                         RET

4173 DD7E00             XOR1:   LD A,(IX+0)
4176 4F                         LD C,A
4177 CDAA41                     CALL LOADR
417A DD7E00                     LD A,(IX+0)
417D A9                         XOR C
417E DD7700                     LD (IX+0),A
4181 C9                         RET

4182 DD7E00           ANDNOT1:  LD A,(IX+0)
4185 4F                         LD C,A
4186 CDA041                     CALL LDNOTR
4189 DD7E00                     LD A,(IX+0)
418C A1                         AND C
418D DD7700                     LD (IX+0),A
4190 C9                         RET

4191 DD7E00           ORNOT1:   LD A,(IX+0)
4194 4F                         LD C,A
4195 CDA041                     CALL LDNOTR
4198 DD7E00                     LD A,(IX+0)
419B B1                         OR C
419C DD7700                     LD (IX+0),A
419F C9                         RET

                        ; *PROCEDURE LOAD NOT BIT*
41A0 CDAA41           LDNOTR:   CALL LOADR
41A3 2F                         CPL
41A4 E601                       AND 01
41A6 DD7700                     LD (IX+0),A
41A9 C9                         RET

                        ; *PROCEDURE LOAD BIT*
41AA DB31             LOADR:    IN A,(031H)     ;NUMBER INPUT PORT ON 8255
41AC 57                         LD D,A          ;DATA INPUT STORE REGISTER I
41AD 23                         INC HL
41AE 7E                         LD A,(HL)
41AF E60F                       AND 00FH
41B1 3D               LDR4:     DEC A
41B2 FE00                       CP 00H
41B4 2804                       JR Z,LDR3
41B6 CB0A                       RRC D
41B8 18F7                       JR LDR4
41BA 7A               LDR3:     LD A,D
41BB E601                       AND 01H   ;CLEAR BIT 1 TO BIT 7-DATA_IN BIT
41BD DD7700                     LD (IX+0),A
41C0 C9                         RET
0000                            END
```

PAGE: 1 B:CONVT111.P

```
program converThexfile;
uses crt,twwindow,twdir,file_Os;
Type    byterecord=record
            databyte:byte;

         end;

        fivarrec = file of byterecord;
        fivarobj=fivarb ;
        line=string;
        optype=string;
        hexline=string[57];
        hexword=string[2];
        hexbyte=byte;
Var     filein:fivart;
        fileout:fivarrec;
        hexln:hexline;
        hexbte:hexbyte;
        byterec:byterecord;
        ln:line;
        opcode:optype;
        x,count:integer;
        pause:boolean;
        inputfile,outputfile,word,application:string;

procedure readfileconvert(var fin:fivart;var fout:fivarobj);
label start;
  begin
start:    clrscr;
          write(' enter name to input textfile ?  ');
          readln(readname); calldir(readname,drive,selectfi);
          write(' ennter name to output objfile ?  ');
          readln(outputfile);
          writeln(' strike any key when ready');wait:=readkey;
          writeln;
          assign(fin,selectfi);
          assign(fout,outputfile);
          if  exist(fin) then
           begin
           reset(fin);
           rewrite(fout);
          end
         else
           begin
             writeln('No file  ',inputfile);
             if chkquit then  exit
              else goto start;
           end
    end;

procedure convert_string2_to_DECI(var h:hexword; var Dbyte:byte);
var deci: array[1..2] of byte ;
    m,rs,vl: integer;
    s:string;

begin
                for m:= 1 to 2 do begin
                   case h[m] of
                      'A'       : DECI[m] := 10 ;
                      'B'       : DECI[m] := 11 ;
                      'C'       : DECI[m] := 12 ;
                      'D'       : DECI[m] := 13 ;
                      '0'..'9'  : val(h[m],deci[m],rs);
{                      '1'..'9'  : val(h[m],deci[m],rs);}
{                      '2'       : deci[m] := 2 ;
                      '3'       : DECI[m] := 3 ;
                      '4'       : DECI[m] := 4 ;
                      '5'       : DECI[m] := 5 ;
                      '6'       : DECI[m] := 6 ;
```

```
                    '7'       : DECI[m] := 7 ;
                    '8'       : DECI[m] := 8 ;
                    '9'       : deci[m] := 9 ;
                    'A'       : deci[m] := 10 ;
                    'B'       : DECI[m] := 11 ;
                    'C'       : DECI[m] := 12 ;
                    'D'       : DECI[m] := 13 ;
                    'E'       : DECI[m] := 14 ;
                    'F'       : DECI[m] := 15 ;}
               end;
               Dbyte := deci[1]*10+deci[2] ;
end;
END;

procedure convert_string2_to_hex(var h:hexword; var hbyte:byte);
var deci: array[1..2] of byte ;
    m,rs,vl: integer;
    s:string;

begin
                for m:= 1 to 2 do begin
                   case h[m] of
                     '0'..'9' : val(h[m],DECI[m],rs);
                      '1'       : DECI[m] := 1 ;
{                   '2'       : deci[m] := 2 ;
                    '3'       : DECI[m] := 3 ;
                    '4'       : DECI[m] := 4 ;
                    '5'       : DECI[m] := 5 ;
                    '6'       : DECI[m] := 6 ;
                    '7'       : DECI[m] := 7 ;
                    '8'       : DECI[m] := 8 ;
                    '9'       : deci[m] := 9 ; }
                    'A'       : deci[m] := 10 ;
                    'B'       : DECI[m] := 11 ;
                    'C'       : DECI[m] := 12 ;
                    'D'       : DECI[m] := 13 ;
                    'E'       : DECI[m] := 14 ;
                    'F'       : DECI[m] := 15 ;
                end;
                hbyte := deci[1]*16+deci[2] ;
end;
END;

procedure convert_to_purehex(var hln:hexline;var hexb:hexbyte);
var  opt: hexword;
     j,k,countopt:byte;
     hexnum:byte;
          { optempt:hex format;}
      begin
            pause:=false;
              if hln[1] =':' then begin
            opt:= copy(hln,2,2);
            convert_string2_to_DECI(opt,countopt);
            if countopt >0 then begin
              k:=10;
              for j:=1 to countopt do begin
                 opt:= copy(hln,k,2);
                 convert_string2_to_hex(opt,hexnum);
                 k:= k+2 ;
                 hexb:=HEXNUM;

              end;
            end;
      end;

end;

begin  clrscr; { MAIN }
  application:='reset';
  readfiletext(filein,application);  inputfile:=application;
```

```
  application:='rewrite';
  readln(outputfile);
  assign(fileout,outputfile);
  rewrite(fileout);
  pause:=false;
  while not ((eof(filein))or(pause=true)) do begin
    hexln:='';
    readln(filein,hexln);
    if copy(hexln,8,2) = '01' then pause :=true ;
    convert_to_purehex(hexln,hexbte);    { hexfile must be adjust
    byterec.databyte:=hexbte;
    write(fileout,byterec);
  end;
  close(filein);  close(fileout);
  writeln('complete the  ',inputfile,'file ');

end.
```

```
PROGRAM READDAT(INPUT,OUTPUT);
USES DOS,CRT,twwindow,twdir;
VAR  COMFILE,CODFILE:STRING[15];
     RE:TEXT;
     WR:FILE OF BYTE;
     LINE,INT,N:INTEGER;
     CODE,CODE1,CHAR1:BYTE;
     C,CH:CHAR;
     DATA:STRING[7];
     REGS: REGISTERS;
     CHECKLINE1,CHECKERR,CHECKOUT,CHECKLINE3,MARK:BOOLEAN;
     editfile,drive,selectfi:string;
PROCEDURE SETWINDOW(x1,y1,x2,y2:integer);
     VAR   I1:INTEGER;
     CONST UPLEFT=#201;
           UPRIGHT=#187;
           LOLEFT=#200;
           LORIGHT=#188;
           HORBAR=#205;
           VERBAR=#186;
BEGIN
     WINDOW(X1-1,Y1-1,X2+1,Y2+1);CLRSCR;
     WINDOW(1,1,80,25);
     GOTOXY(X1-1,Y1-1);
     WRITE(UPLEFT);
    FOR I1:=X1 TO X2 DO
     WRITE(HORBAR);
    WRITE(UPRIGHT);
    FOR I1:=Y1 TO Y2 DO
       BEGIN GOTOXY(X1-1,I1);WRITE(VERBAR);
             GOTOXY(X2+1,I1);WRITE(VERBAR);END;
     GOTOXY(X1-1,Y2+1);WRITE(LOLEFT);
    FOR I1:=X1 TO X2 DO
       WRITE(HORBAR);
     WRITE(LORIGHT);GOTOXY(X1,Y1+1);
    FOR I1:=X1 TO X2 DO WRITE(HORBAR);
     GOTOXY(2,2);WRITE('CTRL: ');
     GOTOXY(7,2);WRITE('F1':15,'F2':17,
     'F3':15);
     GOTOXY(7,2);WRITE('(Code)':12);
     GOTOXY(23,2);WRITE('(Control)':13);
     GOTOXY(39,2);WRITE('(Send)':11);
     GOTOXY(61,2);WRITE('ESC-QUIT':10);
     TEXTCOLOR(5);
     X1:=7;Y1:=2;
    WHILE X1<60 DO
    BEGIN
      I1:=0;
     WHILE I1 < 4 DO
      BEGIN GOTOXY(X1,Y1);
      WRITE(#219);X1:=X1+1;I1:=I1+1;END;
      X1:=X1+12;
      END;
      WINDOW(2,4,79,23);
   END;

PROCEDURE SENDCH ( OUTCH :INTEGER);
BEGIN
 REPEAT
 UNTIL ( ( PORT[ $3fd] AND  $20) <> 0 );
 PORT[$3F8] :=  OUTCH;
END;

PROCEDURE READCH(VAR READ:INTEGER);
BEGIN
 REPEAT
 UNTIL ( ( PORT[$3fd] AND  1) <> 0 ) ;
 READ := PORT[$3f8];
END;
```

```
PROCEDURE REKEY(K:CHAR);
        BEGIN
            REGS.AH:=0;  REGS.AL:=$87;
            INTR($14,REGS);
            SENDCH(ORD(K));TEXTCOLOR(19);
            IF UPCASE(K)='R' THEN WRITELN('OK.sent R(un) key');
            IF UPCASE(K)='S' THEN WRITELN('OK.sent S(top) key');
            TEXTCOLOR(5);
        END;
PROCEDURE CONKEY ;
VAR CH:CHAR;
BEGIN
  CH:='0';
WHILE CH<>'Q' DO
BEGIN
WRITE('Run file now (R(un) or S(top) or Q(uit mode)key) ?');
                CH:=READKEY;WRITELN(CH);
                IF UPCASE(CH)='Q'THEN EXIT;
                IF (UPCASE(CH)='R')OR(UPCASE(CH)='S') THEN
                REKEY(CH)
                ELSE
                WRITELN('Error keypress!!');
END;
END;

PROCEDURE COMNUM;
   BEGIN
    READ(RE,C);
    WHILE C=' ' DO READ(RE,C);
    IF CODE IN [74..75] THEN
       IF C=';' THEN
          BEGIN READLN(RE);LINE:=LINE+1;N:=N+1;
          MARK:=FALSE;EXIT;END
       ELSE
       BEGIN  WRITELN;
       WRITELN('No bit number in command line ',LINE,
          ' or ";" expected');
          CHECKLINE3:=TRUE;EXIT;END;
    IF (C<='0')OR(C>='9')OR(C=';')THEN
       BEGIN
       WRITELN;
       WRITELN('Error out of span integer bit in line"',LINE);
       CHECKLINE3:=TRUE;EXIT;END;
          CASE C OF
          '1':CODE:=49;'2':CODE:=50;'3':CODE:=51;'4':CODE:=52;
          '5':CODE:=53;'6':CODE:=54;'7':CODE:=55;'8':CODE:=56;
          END;
          WRITE(CODE,' ');WRITE(WR,CODE);
          READ(RE,C);
           WHILE C=')' DO
            BEGIN
           IF CHECKOUT THEN
            BEGIN WRITELN;
            WRITELN('No "OUT *);"  form program plc line ',LINE)
            CHECKLINE3:=TRUE;EXIT;END;
            CODE1:=ORD(C);N:=N-1;
            WRITE(WR,CODE1);
            WRITE(CODE1,' ');
            READ(RE,C);END;
          IF C<>';' THEN
          BEGIN WRITELN('Error statement ";" in line ',LINE);
          CHECKLINE3:=TRUE;EXIT;END;
          READLN(RE);LINE:=LINE+1;
   END;
```

```
PROCEDURE OUTSERI;
          VAR SEN:BYTE;CO:FILE OF BYTE;
      BEGIN
           CODFILE:='';
           WHILE CODFILE='' DO
           BEGIN
           WRITE('Name file obcode ?');
           READLN(CODFILE);
           END;
           ASSIGN(CO,CODFILE);
           {$I-}RESET(CO);{$I+}
           IF IORESULT<>0 THEN
           BEGIN
             WRITELN(#12'.Obcode file not found.'#11);
             EXIT;
           END;
           RESET(CO);
           WHILE NOT(EOF(CO)) DO
           BEGIN
             READ(CO,SEN);
             IF NOT(SEN IN [65..76]) AND NOT(SEN IN [49..56])AND
             (SEN<>$29) THEN
             BEGIN WRITELN;WRITELN('Not complete send code file',
             ' or error code');CHECKERR:=FALSE;CLOSE(CO);
             EXIT;END;
             REGS.AH:=0;
             REGS.AL:=$87;
             {10000111,1200BAUD,NOPARITY,
             2-STOP BIT,8 CHARACTER}
             SENDCH(SEN);WRITE(CHR(SEN),' ');
           END;
           IF SEN<>69 THEN
           BEGIN WRITELN;
             WRITELN('Error end code,please set new hardware');
             CHECKERR:=FALSE;EXIT; END;
             TEXTCOLOR(19);
             WRITELN;WRITELN(#7'OK.for sent obcode file');
             TEXTCOLOR(5);CHECKERR:=FALSE;CLOSE(CO);
           END;

PROCEDURE SETSTRING;
BEGIN
  IF (DATA<>'LD')AND(DATA<>'OR')AND(DATA<>'AND')
  AND(DATA<>'OUT')AND(DATA<>'OUTNOT')
  AND(DATA<>'LDNOT')AND(DATA<>'ANDNOT')
  AND(DATA<>'ORNOT')AND(DATA<>'XOR')
  AND(DATA<>'AND(')AND(DATA<>'OR(')
  AND(DATA<>')')AND(DATA<>'END')  THEN
  BEGIN
    WRITELN;
    WRITE('Error statement..');TEXTCOLOR(19);
    WRITE(DATA);TEXTCOLOR(5);
    WRITELN('..For compile in line" ',LINE,
    ' or ";" statement');
    CHECKLINE1:=TRUE
    END;
  IF NOT(MARK) THEN
  BEGIN
    IF DATA='LD'       THEN
    BEGIN CODE:=65 ;MARK:=TRUE;EXIT;END;
    IF DATA='LDNOT'    THEN
    BEGIN CODE:=70 ;MARK:=TRUE;EXIT;END;
    WRITELN;
    WRITELN('No compleat form program plc in line ',LINE);
    CHECKLINE1:=TRUE;EXIT;
    END;
    IF DATA='AND'      THEN  CODE:=66 ;
    IF DATA='OR'       THEN  CODE:=67 ;
    IF DATA='OUT'      THEN
    BEGIN CODE:=68; CHECKOUT:=TRUE; END;
```

```
    IF DATA='OUTNOT'   THEN
    BEGIN CODE:=76; CHECKOUT:=TRUE; END;
    IF DATA='ANDNOT'   THEN  CODE:=71 ;
    IF DATA='ORNOT'    THEN  CODE:=72 ;
    IF DATA='XOR'      THEN  CODE:=73 ;
    IF DATA='AND('     THEN  CODE:=74 ;
    IF DATA='OR('      THEN  CODE:=75 ;
    IF (DATA='LD')OR(DATA='LDNOT') THEN
    BEGIN WRITELN;
    WRITELN('You can',#39,'t load in this line ',LINE);
    CHECKLINE1:=TRUE;END;
  END;

PROCEDURE CODEFILE;
BEGIN
  EDITFILE:='';
  WHILE EDITFILE='' DO
  BEGIN WRITE('Name file edit ?');
  READLN(EDITFILE);
  calldir(editfile,drive,selectfi);
  END;
  IF CHECKERR=FALSE THEN
  BEGIN
    ASSIGN(RE,EDITFILE);
    {$I-}RESET(RE);{$I+}
    IF IORESULT<>0 THEN
    BEGIN
      WRITELN(#12'.Edit file not found.'#11);
      EXIT;
    END;
 END;
 RESET(RE);
 COMFILE:='';
 WHILE COMFILE=''DO
 BEGIN WRITE('Obcode file name ?');
   READLN(COMFILE);END;
   IF CHECKERR=FALSE THEN
   BEGIN
     ASSIGN(WR,COMFILE);CHECKERR:=TRUE;
   {$I-}RESET(WR);{$I+}
     IF IORESULT<>0 THEN
     BEGIN
       WRITELN(#12'.Obcode file open on disk ',
       'and new call',#39,COMFILE,#39,'file'#11);
       EXIT;
     END;
  END;
  REWRITE(WR); LINE:=1;N:=0;
  CHECKLINE1:=FALSE;CHECKLINE3:=FALSE;
  MARK:=FALSE;CHECKOUT:=FALSE;DATA:='';
  WHILE NOT(EOF(RE)) DO
  BEGIN
    READ(RE,C);
    WHILE (C=' ') OR (C=';') DO
    BEGIN
      IF C=';'THEN
      BEGIN LINE:=LINE+1;READLN(RE);END;
      READ(RE,C);
      END;
   DATA:=UPCASE(C);INT:=1;
   WHILE C<>' ' DO
     BEGIN
     READ(RE,C);
     IF C<>  ' THEN
     BEGIN
     INT:=INT+1;
     C:=UPCASE(C);
     DATA:=DATA+C;
     END;
   IF DATA='END'THEN
```

```
    BEGIN
      IF EOLN(RE) THEN READLN(RE);
      READ(RE,C);CODE:=69;WRITE(CODE);
      WRITE(WR,CODE);
      REPEAT
      BEGIN
        IF EOLN(RE) THEN READLN(RE);
        READ(RE,C);
        IF (C<>' ') AND (C <> #26) THEN
        BEGIN
          WRITELN;
          WRITELN('Error in edit ',EDITFILE,' file END');
          CHECKERR:=FALSE;CLOSE(RE);CLOSE(WR);EXIT;
          END;
       END;
       UNTIL EOF(RE);
       WRITELN;
    IF N<>0 THEN
    BEGIN WRITELN('No compleat program ")" loop');
      EXIT;END;
      WRITELN(DATA);TEXTCOLOR(19);
      WRITELN(#7'Ok. end of compile code');
      TEXTCOLOR(5);
    CHECKERR:=FALSE;
    CLOSE(RE);
    CLOSE(WR);EXIT;
  END;
  IF (INT>7)THEN
  BEGIN
    WRITELN;
    WRITELN('Can not compile in line" ',LINE);
    CHECKERR:=FALSE;
    CLOSE(RE);CLOSE(WR);EXIT;
    END;
    END;
    SETSTRING;
    IF NOT(CHECKLINE1) THEN
    BEGIN WRITE(CODE,' ');WRITE(WR,CODE);
      COMNUM;END;
      IF CHECKLINE3 OR CHECKLINE1 THEN
      BEGIN
        CHECKERR:=FALSE;
        CLOSE(RE);CLOSE(WR);
        EXIT;
        END;
      IF EOF(RE) THEN
      BEGIN
        WRITELN;
        WRITELN('Error in edit file END');
        CHECKERR:=FALSE;CLOSE(RE);CLOSE(WR);
        EXIT;
        END;
       END;
     END;

BEGIN (*MAIN PROGRAM*)
  SETWINDOW(2,2,79,23);
  REGS.AH:=0;INTR($16,REGS);
  CHAR1:=ORD(REGS.AH);CHECKERR:=FALSE;
  WHILE CHAR1<>1  DO
  BEGIN
    IF (CHAR1 IN [$5E..$60]) OR (CHAR1=46) THEN
    BEGIN
      CASE CHAR1 OF
        $5E   : CODEFILE;{CODE FILE}
        $60   : OUTSERI;{SENT FILE}
        46    : CLRSCR;{CLEAR SCREEN}
        $5F   : CONKEY;{CONTROL KEY}
        END
    END
```

```
    ELSE
      BEGIN TEXTCOLOR(5);
      WRITELN(#174'press CTRL-FUNCTION(1-3)',
      'or Esc key!! or ALT-C(lear screen)..');
      END;
    REGS.AH:=0;INTR($16,REGS);
    CHAR1:=ORD(REGS.AH);
    END;
    WINDOW(1,1,80,25);
    CLRSCR;TEXTCOLOR(5);WRITELN(#14,'goodbye..bye..bye..'#6#3);
END
```

ภาคผนวก ง OP CODE TABLE

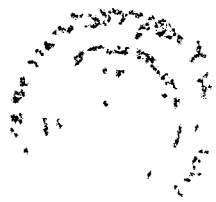
obcode table

HEX	DEC	COMMENT
01	65	'LD'
02	66	'AND'
43	67	'OR'
44	68	'OUT'
45	69	'END'
46	70	'LDNOT'
47	71	'ANDNOT'
48	72	'ORNOT'
49	73	'XOR'
4A	74	'AND('
4B	75	'OR('
4C	76	'OUTNOT'
29	41	')'
31	49	'1'
32	50	'2'
33	51	'3'
34	52	'4'
35	53	'5'
36	54	'6'
37	55	'7'
38	56	'8'

ตารางแปลง จากคำสั่ง เป็น opcode plc

กิตติกรรมประกาศ

ด้วยเหตุที่โครงงาน PLC ได้สำเร็จลุลวงไปด้วยดีเนื่องด้วย อ.สุเธียร เกียรติสุนทร ได้ให้คำปรึกษา และแนะนำบุคคล คือ พี่วิกิจ ที่ได้เอื้อเฟื้ออุปกรณ์สำคัญบางส่วนของโครงงาน และข้าพเจ้าขอแสดงความขอบคุณ คณาจารย์ทุกท่าน และทุกคนที่ช่วยข้าพเจ้าให้มีสถานที่และเวลาในการศึกษาในโครงงานนี้



เอกสารอ้างอิง

1). Coffron, J.W. , Z-80 APPLICATION, Sybex , 1983 .

2). Zaks, R. , PROGRAMMING THE Z-80 ,Sybex Inc , 1980.

3). " IBM TECHNICAL REFERENCE " IBM Personal Computer Hard reference Library , 1983 .

4). Turbo Pascal 5.5 , User Guide & Reference Guide .

5). " สุรศักดิ์ สงวนไทย " แอดวานซ์เทอร์โบปาสคาล เวอร์ชั่น 4.0 " , ซีเอ็ดยูเคชั่น , 2521 .