ปฏิบัติการทดลองที่ 2

เสนอ

ผู้ช่วยศาตราจารย์.ดร.พนัส นัถฤทธิ์

จัดทำโดย

58364876 นายอาทิตย์ แซ่ว่าง

58366450 นายศิวศิษฎ์ สารขาว

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา 305281 ไมโครโพรเซสเซอร์และภาษาแอสเซมบลี

ภาคเรียนที่ 1/ปีการศึกษา 2560

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สาขาวิชาวิศกรรมคอมพิวเตอร์

มหาวิทยาลัยนเรศวร

คำนำ

รายงาน ปฏิบัติการทดลองที่ 2 ซึ่งเป็นส่วนหนึ่งของรายวิชา 305381 ไมโครโพรเซสเซอร์และ ภาษาแอสเซมบลี (Microprocessor and Assembly Language) จัดทำขึ้น เพื่อศึกษาการเขียนโปรแกรมภาษาแอสแซมบลีเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 ผ่าน LED

คณะผู้จัดทำหวังว่ารายงานเล่มนี้ จะสามารถเป็นประโยชน์ ต่อผู้ที่ต้องการศึกษาค้นคว้าข้อมูล ได้ตรงตามวัตถุประสงค์ หากรายงานนี้มีข้อผิดพลาดประการใด ผู้จัดทำขออภัยมา ณ ที่นี้

คณะผู้จัดทำ

นายอาทิตย์ แซ่ว่าง

นายศิวศิษฎ์ สารขาว

บทนำ

ไมโครคอนโทรลเลอร์ (Microcontroller) หมายถึง ไมโครโปรเซสเซอร์ตัวหนึ่งที่มี หน่วยความจำชั่วคราว (RAM : Random Access Memory) หน่วยความจำถาวร (ROM : Read Only Memory) หน่วยรับ/ส่งข้อมูลจากภายนอก (Input/Output Port) อยู่ภายในตัวมันเพียงตัวเดียว (Single Chip) ซึ่งไมโครโปรเซสเซอร์ต้องอาศัยหน่วยความจำและหน่วยอินพุต/เอาต์พุตภายนอก ฉะนั้นไมโครคอนโทรลเลอร์จึงมีข้อดีกว่าไมโครโปรเซสเซอร์มาก ในเรื่องของความประหยัดและความสะดวกในการใช้งาน ปัจจุบันจึงนิยมใช้ไมโครคอนโทรลเลอร์ มากกว่าโดยเฉพาะในงานควบคุมทางด้านอุตสาหกรรม และเครื่องใช้ไฟฟ้าในชีวิตประจำวัน เช่น การควบคุมอุณหภูมิ การควบคุมหุ่นยนต์ เครื่องซักผ้าแบบโปรแกรมได้ การควบคุมการปิด-เปิดไฟฟ้าในอาคารการควบคุมไฟวิ่ง เป็นต้น

ในปัจจุบันผู้ผลิตได้พัฒนาให้ไมโครคอนโทรลเลอร์สามารถทำงานได้เร็วขึ้นโดยเพิ่มความสามารถในการรองรับคริสตอลความถี่ที่สูงขึ้น รวมไปถึงการปรับปรุงการทำงานภายในให้ไมโครคอนโทรลเลอร์ใช้จำนวนสัญญาณนาฬิกาในการสร้างแมชชีนไซเคิลน้อยลง โดยในบางรุ่น 1 แมชชีนไซเคิลใช้สัญญาณนาฬิกาเพียงแค่ 1 ลูกเท่านั้น

ซึ่งการทดลองสำหรับวิชา 305381 ไมโครโพรเซสเซอร์และภาษาแอสเซมบลี (Microprocessor and Assembly Language) จะใช้ไมโครคอนโทรลเลอร์ตระกูลMCS51 ในการทดลอง ใช้ภาษา Assembly ในการโปรแกรมควบคุมไมโครคอนโลเลอร์ และใช้งานโปรแกรม Flash Magic สำหรับดาวน์โหลดโปรแกรมลงบอร์ดไมโครคอนโทรลเลอร์ การทดลองนี้เป็นการทดลอง เกี่ยวกับการใช้งานเบื้องต้น สำหรับใช้ไมโครคอนโทรลเลอร์ตระกูล MCS51 89V51RD2 เป็นการทดลองการใช้พอร์ท P0-P3 เป็นอินพุต - เอาท์พุตพอร์ท โดยการใช้ภาษาแอสเซมบลีในการเขียนโปรแกรมคำสั่งในการทดลองต่างๆ และเรียนรู้การดาวน์โหลดโปรแกรมที่ผู้ทดลองเขียนขึ้นลงในตัว ไมโครคอนโทรลเลอร์ MCS51 89V51RD2

วัตถุประสงค์ของการทดลอง

1. เพื่อเรียนรู้การทำงานของ Microcontroller MCS-51
2. เพื่อเรียนรู้วิธีการใช้งานโปรแกรม Flash Magic สำหรับดาวน์โหลดโปรแกรมลงบอร์ดไมโครคอนโทรลเลอร์
3. เพื่อศึกษาการเขียนโปรแกรม rotate
4. เพื่อศึกษาการเขียนโปรแกรม Delay

เนื้อหาและทฤษฎี

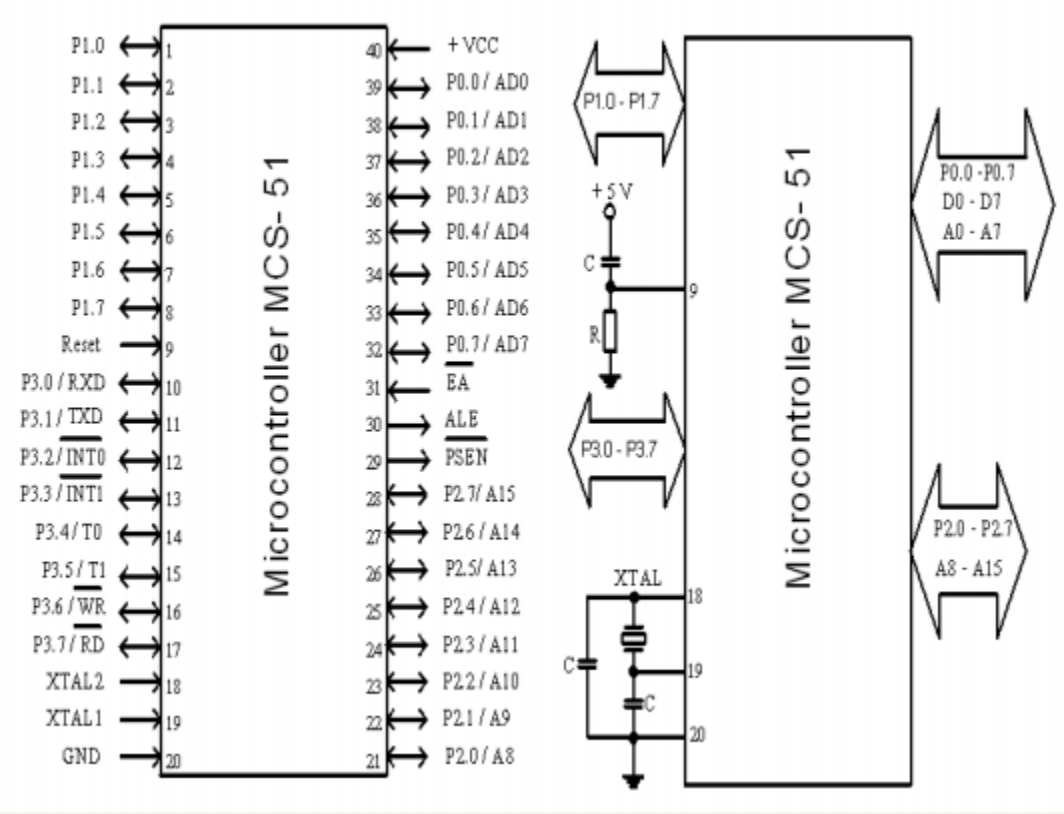
<u>ไมโครคอนโทรลเลอร์ MCS-51</u>

มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 2 ตัว คือไทเมอร์ 0 และไทเมอร์ 1 ส่วนในไมโครคอนโทรลเลอร์อนุกรม 8052 ซึ่งออกมาทีหลังจะมีไทเมอร์/เคาน์เตอร์ 3 ตัว นั้นคือมีไทเมอร์ 2 เพิ่มขึ้นมา ไทเมอร์/เคาน์เตอร์แต่ละตัวสามารถเลือกใช้งานเป็นไทเมอร์ หรือเคาน์เตอร์ก็ได้ และทำงานได้อย่างเป็นอิสระต่อกัน ในการทำงานเป็นไทเมอร์นั้นจะใช้หลักการเพิ่มค่ารีจิสเตอร์ไทเมอร์ ทุก ๆ Machine Cycle ซึ่งมีช่วงเท่ากับ 12 คาบสัญญาณนาฬิกาที่ถูกสร้างขึ้นจากคริสตอลที่ต่อใช้งานให้กับไมโครคอนโทรลเลอร์นั้นเอง สำหรับบทความนี้จะอธิบายรายละเอียดและตัวอย่างการใช้งานไทเมอร์ 0 และ 1 เท่านั้นนะครับ สำหรับไทเมอร์ 2 สามารถอ่านรายละเอียดได้ที่บทความการใช้พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

โครงสร้างพื้นฐานที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล 8051 ซึ่งมีรายละเอียดดังนี้

1. เป็นไมโครคอนโทรลเลอร์ที่มีหน่วยประมวลผลกลางแบบ 8 บิต
2. มีคำสั่งคำนวณทางคณิตศาสตร์ และตรรกศาสตร์ (Boolean processor)
3. มีแอดเดรสบัสขนาด 16 บิตทำให้สามารถอ้างตำแหน่งหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้ 64 กิโลไบต์
4. มีหน่วยความจำ (RAM) ภายในขนาด 128 ไบต์ (8051/8031) หรือ 256 ไบต์ (8052/8032)
5. มีพอร์ตอนุกรมทำงานแบบดูเพล็กซ์เต็ม (Full Duplex) 1 พอร์ต
6. มีพอร์ตอินพุต/เอาต์พุตแบบขนานจำนวน 32 บิต
7. มีไทเมอร์ 2 ตัว (8051/8031) หรือ 3 ตัว (8052/8032)
8. มีวงจรควบคุมการเกิดอินเตอร์รัปต์ 5 ประเภท (8051/8031) หรือ 6 ประเภท (8052/8032)

9. มีวงจรออสซิลเลเตอร์ภายในตัว



<u>ไมโครคอนโทรลเลอร์ MCS-51</u>

ประกอบด้วยพอร์ตที่ใช้เป็นอินพุต/เอาต์พุตเพื่อติดต่อกับอุปกรณ์รอบนอกได้ 4 พอร์ต คือP0, P1, P2 และ P3 มีรายละเอียดแต่ละพอร์ตดังนี้

-P0 (P0.0-P0.7) เป็นอินพุตหรือเอาต์พุตพอร์ตถ้ามีการขยายหน่วยความจำภายนอกหรืออินพุต/ เอาต์พุตพอร์ตภายนอกจะใช้เป็น Data Bus (D0-D7) และ Address Bus (A0-A7)

-P1 (P1.0-P1.7) เป็นอินพุตหรือเอาต์พุตพอร์ต

-P2 (P2.0-P2.7) เป็นอินพุตหรือเอาต์พุตพอร์ต ถ้ามีการขยายหน่วยความจำภายนอกจะใช้เป็น Address Bus (A8-A15)

-P3 (P3.0-P3.7) เป็นอินพุตหรือเอาต์พุตพอร์ต ถ้าไม่ใช้เป็นอินพุต /เอาต์พุตพอร์ตก็สามารถทำหน้าที่ตามชื่อหลังได้ดังนี้

-P3.0/RxD (Receive Data) ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม

-P3.1/TxD (Transmit Data) ใช้เป็นขาเอาต์พุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม

-P3.2/ INT0 (Interrupt 0) รับสัญญาณขัดจังหวะจากภายนอก No. 0

-P3.3/ INT1 (Interrupt 1) รับสัญญาณขัดจังหวะจากภายนอก No. 1

-P3.4/T0 (Timer/Counter0) สามารถโปรแกรมได้ว่าจะให้เป็น Timer หรือ Counter ถ้าใช้สัญญาณ Clock จากภายนอกเข้ามาจะเป็น Counter ถ้าใช้สัญญาณ Clock จากภายในจะเป็น Timer

-P3.5/T1 (Timer/Counter1) ทำหน้าที่ทำนองเดียวกับ P3.4/T0

-P3.6/ WR (Write) ส่งสัญญาณควบคุมการเขียนข้อมูลจาก MCS-51ไปยังภายนอก

-P3.7/ RD (Read) ส่งสัญญาณควบคุมการอ่านข้อมูลจากภายนอกเข้ามายัง MCS-51 -Reset เป็นขาอินพุต

<u>การใช้งานโปรแกรม Flash Magic สำหรับดาวน์โหลดโปรแกรมลงบอร์ดไมโครคอนโทรลเลอร์</u>

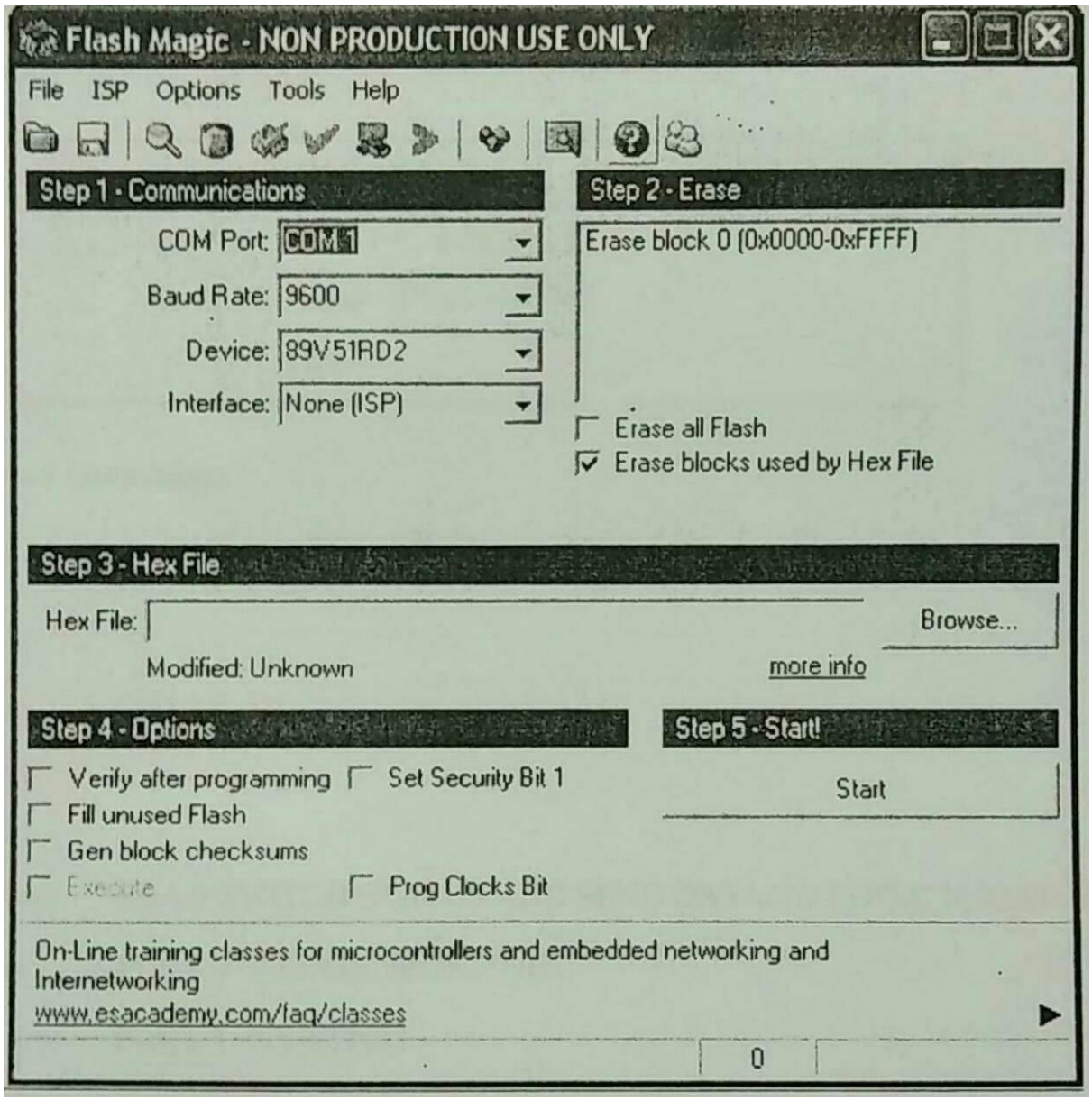
ในการเขียนโปรแกรมภาษาแอสแซมบลีเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์นั้นจะต้องมีการคอมไฟล์โปรแกรมดังกล่าวให้เป็น Hex File แล้วจึงทำการดาวน์โหลดไฟล์นั้นลงบอร์ดไมโครคอนโทรลเลอร์ โดยในการทดลองนี้จะเลือกใช้งานโปรแกรม Flash Magic สำหรับดาวน์โหลดโปรแกรม โดยนิสิตจำเป็นต้องเลือก Option ให้สอดคล้องกับบอร์ดทดลองที่ใช้งานดังนี้

1. เลือกเมนู Options >> Advanced Options ... จะปรากฏหน้าต่าง Advanced Options
2. คลิกที่ TAB ชื่อ Hardware Config แล้วทำการยกเลิกเครื่องหมายถูกที่ปรากฏในช่อง Use DTR to control RST

เมื่อปรับเลือกค่า Option ให้เหมาะสมต่อการใช้งานแล้ว โปรแกรม Flash Magic จะพร้อมสำหรับการโหลดโปรแกรมลงชิปต่อไป โดยมีขั้นตอนดังนี้

- **STEP 1:** เลือกเบอร์ไมโครคอนทรลเลอร์เป็น 89V51RD2 จากนั้นให้กำหนดพอร์ตสื่อสารข้อมูล COM-x ให้ตรงกับพอร์ตสื่อสารของคอมพิวเตอร์ที่ใช้งานอยู่ จากนั้นให้กำหนดค่าความเร็วที่ใช้สื่อสารข้อมูล (ในที่นี้กำหนดให้เป็น 9600 bps ดังแสดงในรูปที่ 1)
- **STEP 2:** เลือกรูปแบบของการลบข้อมูลภายในหน่อยความจำโปรแกรม (Flash Memory) ก่อนที่จะดำเนินการโปรแกรมข้อมูลใหม่ลงไป โดยกำหนดเป็น Erase block used Hex File ซึ่งเป็นการลบข้อมูลเฉพาะ block ที่ต้องการสำหรับการเขียนโปรแกรมข้อมูลใหม่เท่านั้น(โดยจะส่งผลให้การทำงานเร็วกว่าการลบข้อมูลทั้งหมดด้วยคำสั่ง Erase all Flash)

- **STEP 3:** คลิกปุ่ม browse ... เพื่อเปิดหน้าต่าง Select Hex File และเลือกไฟล์ที่ต้องการโปรแกรมลงสู่ไมโครคอนโทรลเลอร์



รูปที่ 1: หน้าต่างของโปรแกรม Flash Magic

- **STEP 4:** เลือก Options การทำงานเพิ่มเติมตามต้องการ
- **STEP 5:** กดปุ่ม Start เพื่อเริ่มขั้นตอนการโปรแกรมลงชิปไมโครคอนโทรลเลอร์

เมื่อปรากฏหน้าต่าง Reset Device ขึ้นมาแล้ว ให้กดปุ่ม RESET บนบอร์ดไมโครคอนโทรลเลอร์ ซึ่งจะเป็นการเริ่มต้นการดาวน์โหลดโปรแกรมลงสู่ชิปทันที โดยจะสามารถสังเกตขั้นตอนการทำงานได้จาก Status bar ที่ขอบด้านล่างของโปรแกรมและเมื่อขั้นตอนการโปรแกรมเสร็จสมบูรณ์ (Finished) ก็ให้กดปุ่ม RESET บนบอร์ดไมโครคอนโทรลเลอร์อีกครั้ง ไมโครคอนโทรลเลอร์จะเริ่มต้นทำงานตามโปรแกรมที่ได้ดาวน์โหลดลงไปใหม่ทันที

การทดลองในรายวิชานี้มีวัตถุประสงค์เพื่อเรียนรู้วิธีการเขียนโปรแกรมภาษาแอสแซมบลีสำหรับใช้ควบคุมการทพงานของไมโครคอนโทรเลอร์ตระกูล MCS-51 โดยเนื้อหาในการทดลองจะประกอบด้วยการทดลองจำนวน 5 การทดลอง ดังนี้

การทดลองที่ 1 – การใช้งานพอร์ท P0-P3 เป็นอินพุต/เอาต์พุตพอร์ท (I/O PORT)

การทดลองที่ 2 – การเขียนโปรแกรมย่อย (SUB – ROUTINE PROGRAM)

การทดลองที่ 3 – การใช้งานคำสั่งกระโดดแบบมีเงื่อนไข (CONDITIONNAL JUMP)

การทดลองที่ 4 – การควบคุมมอเตอร์กระแสตรง (BASIC DC MOTOR CONTROL)

การอ้างถึงตำแหน่งแอดเดรส RL, RLC, RR ทั้ง 3 คำสั่งนี้เป็นคำสั่งในการทำงานการวนบิตบนตัวของแอกคิวมูเลเตอร์ซึ่ง RL เป็นการวนบิตทางซ้าย RR เป็นการวนบิตทางขวา, RLC เป็นการทำการวนทางซ้ายผ่านบิตทด เช่น

RL A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางซ้าย บิต 7 จะ หมุนวนมายังบิต 0

RLC A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางซ้ายผ่าน แฟลกทด โดยบิต 7 จะหมุนไปยังแฟลกทด และข้อมูลของแฟลกทดเดิมจะหมุน เข้ามาในบิต 0

RR A

จำนวนไบต์ : 1

การทำงาน : ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางขวา บิต 0 จะ หมุนวนมายังบิต 7

การเรียกโปรแกรมย่อยการเขียนโปรแกรมประเภทโปรแกรมย่อย (Subroutine) สามารถถูกเรียกขึ้นมาใช้งานด้วยคำสั่ง CALL ใน MCS-51 จะมีอยู่สองคำสั่งคือ LCALL (Long call) และ ACALL (absolute call) โดยทั้งสองคำสั่งต่างกันตรงขนาดของคำสั่ง และระยะห่างของโปรแกรมย่อยที่จะเรียกใช้งาน

LCALL (Long call)

คำสั่งจะมีขนาด 3 ไบต์ โดยไบต์แรกจะเป็นออปโค้ด อีกสองไบต์จะเป็นแอดเดรสที่จะกระโดดไปทำงานโปรแกรมย่อย ซึ่งสามารถเรียกโปรแกรมย่อยได้ในระยะ 64 กิโลไบต์ เมื่อโปรแกรมย่อยถูกเรียกใช้ MCS-51 จะนำค่าแอดเดรสของคำสั่งถัดไปหรือค่า PC เก็บลงในหน่วยความจำสแตกโดยอัตโนมัติ และโหลดค่าแอดเดรสของโปรแกรมย่อยให้กับรีจีสเตอร์ PC และเมื่อทำโปรแกรมย่อยไปจนพบคำสั่ง RET (return) MCS-51 จะกลับไปทำคำสั่งของโปรแกรมที่อยู่ต่อจากคำสั่งเรียกโปรแกรมย่อย โดยคืนค่าที่เก็บอยู่ในสแตกให้กับรีจีสเตอร์

ACALL (absolute call)

คำสั่งเรียกโปรแกรมย่อย ACALL เป็นคำสั่งที่มีขนาด 2 ไบต์ โดยใช้ข้อมูล 11 บิตในคสั่งเป็นค่าแอดเดรส ทำให้สามารถเรียกโปรแกรมย่อยได้ในช่วง 2 กิโลไบต์ การทำงานกับสแตกของคำสั่ง ACALL และ LCALL นี้จะไม่แตกต่างกัน แต่การใช้งานจะต่างกันตรงที่คำสั่ง LCALL เรียกโปรแกรมย่อยได้ในระยะ 64 กิโลไบต์ ส่วน ACALL เรียกโปรแกรมย่อยได้ในช่วง 2 กิโลไบต์

MICROCONTROLLER MCS-51

Summary: SUB-ROUTINE PROGRAM

LAB 2-1

Description: DELAY TIIME SUB-ROUTINE PROGRAM

Hardware: PORT 0 -> LOGIC MONITOR

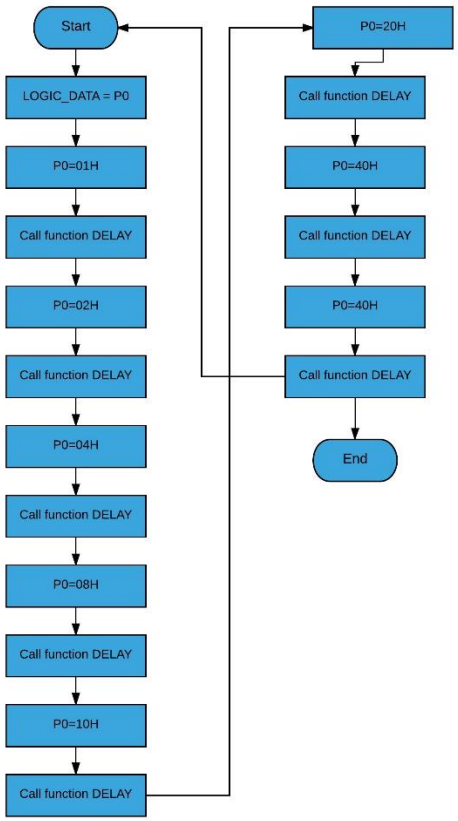
ASM Code:

```
                        ORG     0000H
                        LOGIC_DATA EQU P0
START:          MOV LOGIC_DATA, #01H
                        LCALL DELAY
                        MOV LOGIC_DATA, #02H
                        LCALL DELAY
                        MOV LOGIC_DATA, #04H
                        LCALL DELAY
                        MOV LOGIC_DATA, #08H
                        LCALL DELAY
                        MOV LOGIC_DATA, #10H
                        LCALL DELAY
                        MOV LOGIC_DATA, #20H
                        LCALL DELAY
                        MOV LOGIC_DATA, #40H
                        LCALL DELAY
                        MOV LOGIC_DATA, #80H
                        LCALL DELAY
DELAY:          MOV R5, #5
DELAY0:         MOV R6, #255
DELAY1:         MOV R7, #255
DELAY2:                 DJNZ R7, DELAY2
                        DJNZ R6, DELAY1
                        DJNZ R5, DELAY0
                        RET
                        END
```

วิธีการทดลอง

1. ต่อวงจรโดยต่อ PORT 0 เข้ากับ Logic monitor
2. เขียนโค้ดที่โจทย์กำหนดให้ จากนั้นทำการแปลงไฟล์ให้ได้ไฟล์นามสกุล .hex
3. ทำการรัน ASM Code จากโปรแกรม Flash Magic เพื่อดาวน์โหลดโค้ดลงบอร์ดไมโครคอนโทรลเลอร์ผ่านสายสื่อข้อมูล
4. สังเกตผลการทดลองและบันทึกผลการทดลอง

Flow Chart:



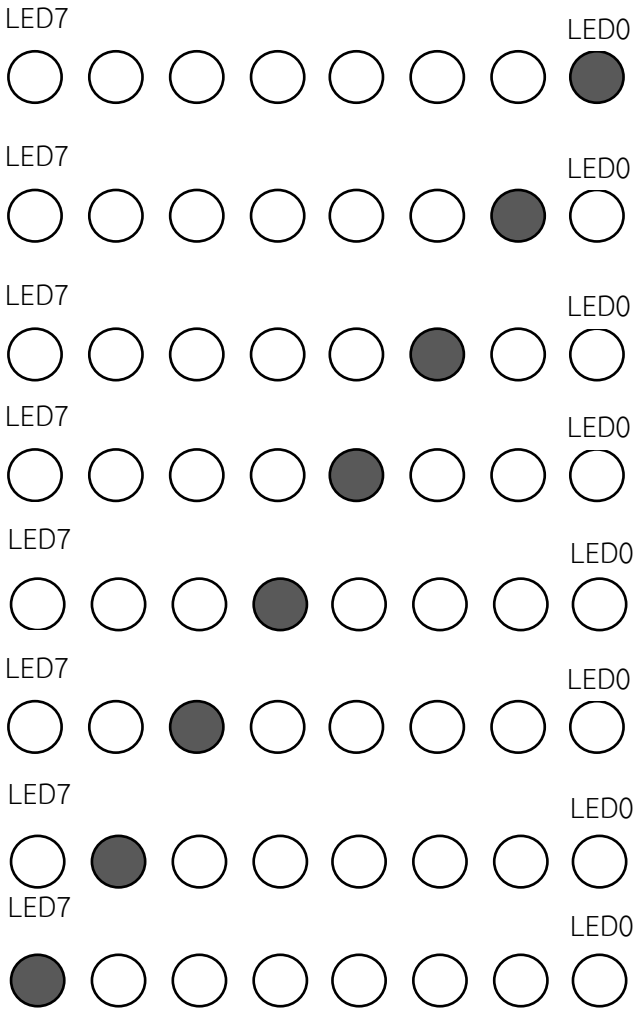
การทำงานของโปรแกรม :

Code	คำอธิบายโปรแกรม
ORG 0000H	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
LOGIC_DATA EQU P0	ประกาศตัวแปร LOGIC_DATA สำหรับเรียกแทนพอร์ต 0
START: MOV LOGIC_DATA, #01H	เริ่มโปรแกมย่อยชื่อ START กำหนดให้ LOGIC_DATA = 01H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #02H	กำหนดให้ LOGIC_DATA = 02H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #04H	กำหนดให้ LOGIC_DATA = 04H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #08H	กำหนดให้ LOGIC_DATA = 08H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #10H	กำหนดให้ LOGIC_DATA = 10H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #20H	กำหนดให้ LOGIC_DATA = 20H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #40H	กำหนดให้ LOGIC_DATA = 40H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
MOV LOGIC_DATA, #80H	กำหนดให้ LOGIC_DATA = 80H เพื่อใช้เป็นตัวชี้ข้อมูล
LCALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
DELAY: MOV R5, #5	โปรแกรมย่อยหน่วงเวลา
DELAY0: MOV R6, #255	โปรแกรมย่อยหน่วงเวลา 0
DELAY1: MOV R7, #255	โปรแกรมย่อยหน่วงเวลา 1
DELAY2: DJNZ R7, DELAY2	โปรแกรมย่อยหน่วงเวลา 2

DJNZ R6, DELAY1	
DJNZ R5, DELAY0	
RET	สิ้นสุดโปรแกรมหน่วงเวลา
END	จบการทำงาน

Results and discussion:

เมื่อป้อนแอสเซมบลีโค้ด โมนิเตอร์จะแสดงดังนี้ คือ ไฟ LED จะติดเป็นสีแดงโดยไล่จาก LED0 – LED7 ไปทีละดวงไปเรื่อยๆ ดังภาพ

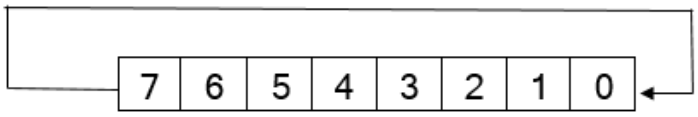


LAB 2-2

Description: ROTATE LEFT DATA IN THE ACCUMULATOR AND DISPLAY ON LOGIC MONITOR AS MOVING LED FROM RIGHT TO LEFT

Hardware: PORT 0 -> LOGIC MONITOR

Instruction: THE **RL** INSTRUCTION RATATES EIGHT BITS IN ACCUMULATOR IN LEFT DIRECTION. BIT 7 OF ACCUMULATOR IS ROTATED IN TO BIT 0, BIT 0 INTO BIT 1, AND SO ON, AS DISPLAYED BELOW. NO FLAGS ARE AFFECTED BY THIS INSTRUCTION



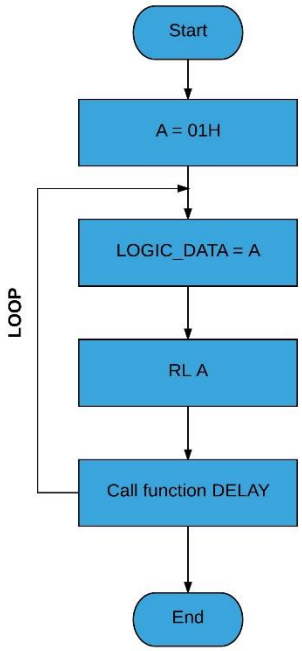
ASM Code:

```
                    ORG     0000H
                    LOGIC_DATA EQU P0
START:      MOV A, #01H
LOOP:       MOV LOGIC_DATA, A
                    RL A
                    LCALL DELAY
                    SJMP LOOP
DELAY:      MOV R5, #5
DELAY0:     MOV R6, #255
DELAY1:     MOV R7, #255
DELAY2:     DJNZ R7, DELAY2
                    DJNZ R6, DELAY1
                    DJNZ R5, DELAY0
                    RET
                    END
```

วิธีการทดลอง

1. เขียนโค้ดที่โจทย์กำหนดให้ จากนั้นทำการแปลงไฟล์ให้ได้ไฟล์นามสกุล .hex
2. ทำการรัน ASM Code จากโปรแกรม Flash Magic เพื่อดาวน์โหลดโค้ดลงบอร์ดไมโครคอนโทรลเลอร์ผ่านสายสื่อข้อมูล
3. สังเกตผลการทดลองและบันทึกผลการทดลอง

Flow Chart:

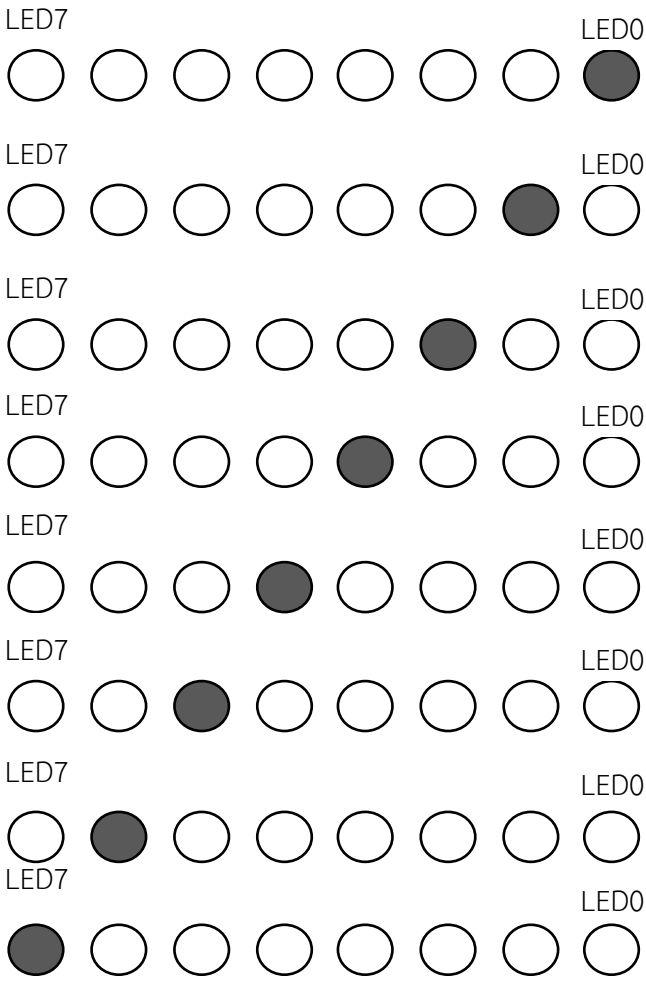


การทำงานของโปรแกรม :

Code	คำอธิบายโปรแกรม
ORG 0000H	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
LOGIC_DATA EQU P0	ประกาศตัวแปร LOGIC_DATA สำหรับเรียกแทนพอร์ต 0
START: MOV LOGIC_DATA, #01H	เริ่มโปรแกมย่อยชื่อ START กำหนดให้ LOGIC_DATA = 0 เพื่อใช้เป็นตัวชี้ข้อมูล
LOOP: MOV LOGIC_DATA	แสดงผล
RL A	หมุนไปทางซ้าย ไม่เก็บค่าบิตทศ
CALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
SIMP LOOP	กลับไปที่ LOOP
DELAY: MOV R5, #5	โปรแกรมย่อยหน่วงเวลา
DELAY0: MOV R6, #255	โปรแกรมย่อยหน่วงเวลา 0
DELAY1: MOV R7, #255	โปรแกรมย่อยหน่วงเวลา 1
DELAY2: DJNZ R7, DELAY2	โปรแกรมย่อยหน่วงเวลา 2
DJNZ R6, DELAY1	
DJNZ R5, DELAY0	
RET	สิ้นสุดโปรแกรมหน่วงเวลา
END	จบการทำงาน

Results and discussion:

เมื่อป้อนแอสเซมบลีโค้ด โมนิเตอร์จะแสดงดังนี้ คือ ไฟ LED จะติดเป็นสีแดงโดยไล่จาก LED0 – LED7 ไปทีละดวงไปเรื่อยๆ ดังภาพ

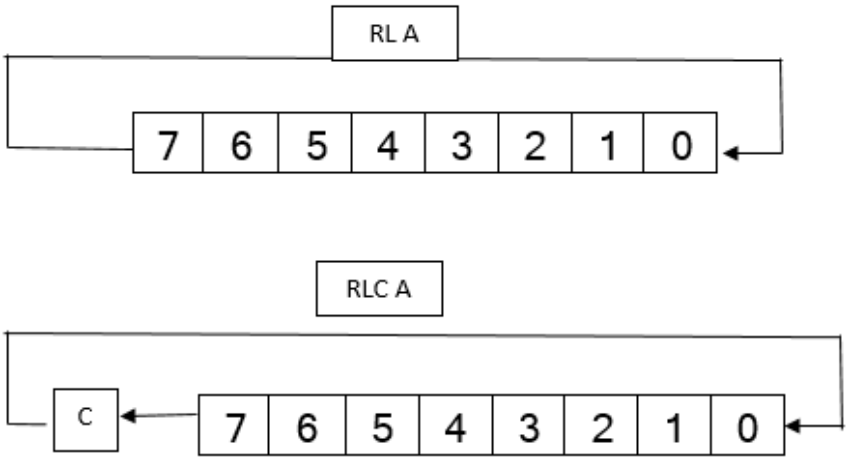


LAB 2-3

Description: ROTATE LEFT DATA THROUGH CARRY FLAG IN ACCUMULATOR AND DISPLAY ON LOGIC MONITOR AS MOVING LED FROM RIGHT TO LEFT

Hardware: PORT 0 -> LOGIC MONITOR

Instruction: THE **RLC** INSTRUCTION ROTATES EIGHT BITS IN ACCUMULATOR AND ONE BIT IN CARRY FLAG IN LEFT DIRECTION. BIT 7 OF ACCUMULATOR IS ROTATED INTO YHE CARRY FLAG WHILE THE VALUE OF CARRY FLAG IS ROTATED INYO BIT 0, BIT 0 OF ACCUMULATOR IS ROTATED BIT 1,BIT 1 INTO BIT 2, AND SO ON, AS DISPLAYED BELOW. NO OTHER FLAGS ARE AFFECTED BY THIS INSTRUCTION



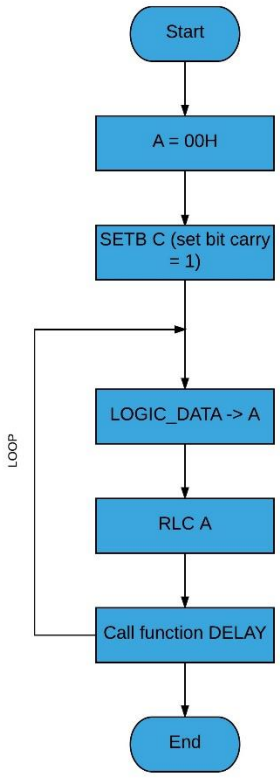
ASM Code:

```
                ORG     0000H
                LOGIC_DATA EQU P0
START:          MOV A, #00H
                SETB C
LOOP:           MOV LOGIC_DATA, A
                RLC A
                LCALL DELAY
                SJMP LOOP
DELAY:          MOV R5, #5
DELAY0:         MOV R6, #255
DELAY1:         MOV R7, #255
DELAY2:         DJNZ R7, DELAY2
                DJNZ R6, DELAY1
                DJNZ R5, DELAY0
                RET
                END
```

วิธีการทดลอง

1. เขียนโค้ดที่โจทย์กำหนดให้ จากนั้นทำการแปลงไฟล์ให้ได้ไฟล์นามสกุล .hex
2. ทำการรัน ASM Code จากโปรแกรม Flash Magic เพื่อดาวน์โหลดโค้ดลงบอร์ดไมโครคอนโทรลเลอร์ผ่านสายสื่อข้อมูล
3. สังเกตผลการทดลองและบันทึกผลการทดลอง

Flow Chart:

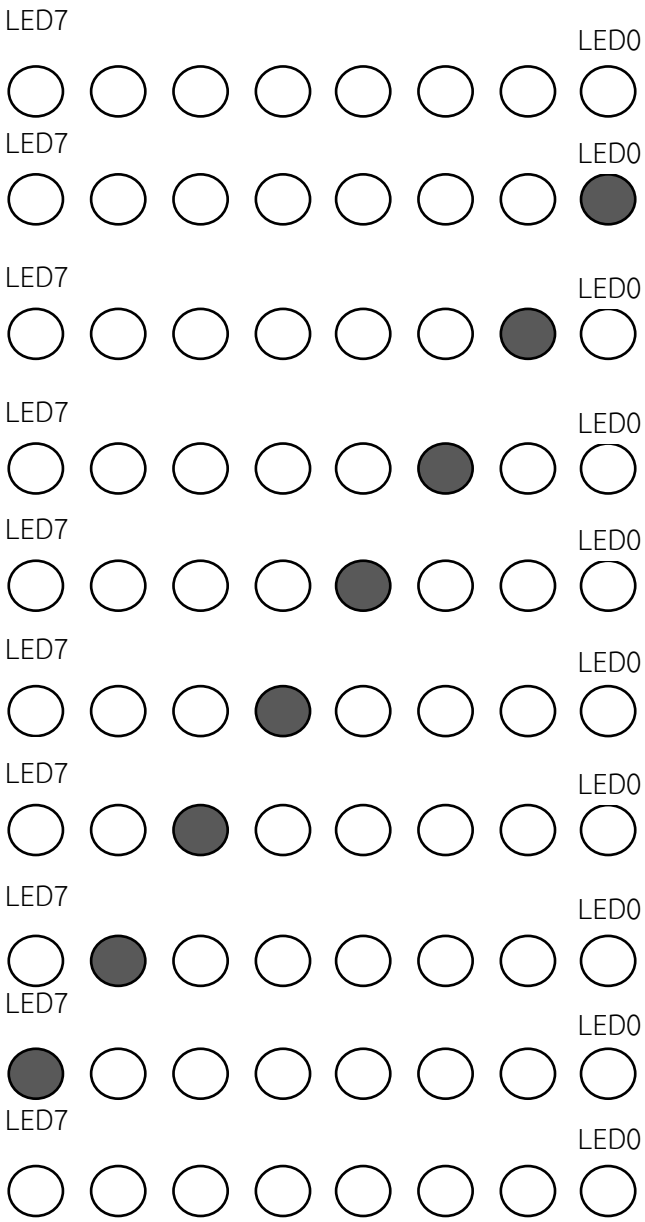


การทำงานของโปรแกรม :

Code	คำอธิบายโปรแกรม
ORG 0000H	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
LOGIC_DATA EQU P0	ประกาศตัวแปร LOGIC_DATA สำหรับเรียกแทนพอร์ต 0
START: MOV LOGIC_DATA, #00H	เริ่มโปรแกมย่อยชื่อ START กำหนดให้ LOGIC_DATA = 0 เพื่อใช้เป็นตัวชี้ข้อมูล
LOOP: MOV LOGIC_DATA, A	แสดงผล
RLC A	หมุนไปทางซ้าย เก็บค่าบิตทศ
CALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
SIMP LOOP	กลับไปที่ LOOP
DELAY: MOV R5, #5	โปรแกรมย่อยหน่วงเวลา
DELAY0: MOV R6, #255	โปรแกรมย่อยหน่วงเวลา 0
DELAY1: MOV R7, #255	โปรแกรมย่อยหน่วงเวลา 1
DELAY2: DJNZ R7, DELAY2	โปรแกรมย่อยหน่วงเวลา 2
DJNZ R6, DELAY1	
DJNZ R5, DELAY0	
RET	สิ้นสุดโปรแกรมหน่วงเวลา
END	จบการทำงาน

Results and discussion:

เมื่อป้อนแอสเซมบลีโค้ด โมนิเตอร์จะแสดงดังนี้ เมื่อเริ่มโปรแกรมไฟดวงที่ 1 จะยังไม่กระพริบในช่วงเวลาแรก LED จะมีการหน่วงเวลาเอาไว้ช่วงเวลาหนึ่ง และจะติดเป็นสีแดงโดยไล่จาก LED0 – LED7 ไปทีละดวงไปเรื่อยๆ ดังภาพ



EXERCISE

1. ให้นิสิตสังเกตการทดลองที่ LAB2-2 และ LAB2-3 ว่ามีความแตกต่างกันอย่างไร เพราะเหตุใด อธิบายมาโดยละเอียด

ตอบ

ASM Code: จาก LAB 2-2

```
                ORG     0000H
                LOGIC_DATA EQU P0
START:          MOV A, #01H
LOOP:           MOV LOGIC_DATA, A
                RL A
                LCALL DELAY
                SJMP LOOP
DELAY:          MOV R5, #5
DELAY0:         MOV R6, #255
DELAY1:         MOV R7, #255
DELAY2:         DJNZ R7, DELAY2
                DJNZ R6, DELAY1
                DJNZ R5, DELAY0
                RET
                END
```

ASM Code: จาก LAB 2-3

```
                ORG     0000H
                LOGIC_DATA EQU P0
START:          MOV A, #00H
                SETB C
LOOP:           MOV LOGIC_DATA, A
                RLC A
                LCALL DELAY
                SJMP LOOP
DELAY:          MOV R5, #5
DELAY0:         MOV R6, #255
DELAY1:         MOV R7, #255
DELAY2:         DJNZ R7, DELAY2
                DJNZ R6, DELAY1
                DJNZ R5, DELAY0
                RET
                END
```

ตอบ

จาก ASM code และผลการทดลองจะสังเกตได้ว่าใน Lab 2-2 แต่จะเริ่มกระพริบในช่วงเวลาเดียวกันกับ Lab2-3 กระพริบไฟครั้งที่สอง หลังจากนั้นจะกระพริบเลื่อนไปทางซ้ายต่อไปเรื่อย ๆ Lab 2-2 มีการตั้งค่าให้ P0 มีค่าเป็น 1 ตั้งแต่แรกทำให้การกระพริบของไฟใน Lab 2-2 จะมีการกระพริบไฟตั้งแต่ครั้งแรกที่โปรแกรมเริ่มทำงาน เมื่อเทียบกับ Lab 2-3 มีการใช้งานคำสั่ง SETB C ซึ่งเป็นการตั้งค่าให้บิต C carry bit เป็น 1 เมื่อเริ่มโปรแกรมไฟดวงที่ 1 จะยังไม่กระพริบในช่วงเวลาแรก

2. ใช้ Lookup table ในการทำงานให้ไฟมีการแสดงผลตามรูปที่ 5.6 ในหนังสือไมโครคอนโทรลเลอร์และการประยุกต์ใช้งานควบคุมหุ่นยนต์ หน้าที่ 70

ตอบ

```
                ORG 0000H
                LED EQU P0
                MOV DPTR, #DATA
START:          MOV R2, #00H
                MOV A, R2
                MOVC A, @A+DPTR
                MOV LED, A
                CALL DELAY
LOOP:           INC R2
                MOV A, R2
                MOVC A, @A+DPTR
                MOV LED, A
                CALL DELAY
                CJNE R2, #06H, LOOP
                SJMP START
DELAY:          MOV R0, #0FH
DELAY1:         MOV R1, #0FFH
DELAY2:         DJNZ R1, DELAY2
                DJNZ R0, DELAY1
                RET
DATA:           DB 81H, 0C3H, 0E7H, 0FFH
                DB 0E7H, 0C3H, 81H
                END
```

Code	คำอธิบายโปรแกรม
ORG 0000H	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
LED EQU P0	ประกาศตัวแปร LED สำหรับเรียกแทนพอร์ต 0

MOV DPTR, #DATA	นำ DPTR ชี้ไปที่ตารางข้อมูล DATA
START: MOV R2, #00H	เริ่มโปรแกมย่อยชื่อ START กำหนดให้ R2 = 0 เพื่อใช้เป็นตัวชี้ข้อมูล
MOV A, R2	นำตัวชี้ข้อมูล R2 มาไว้ที่รีจิสเตอร์ A
MOVC A, @A+DPTR	นำค่าในตาราง DATA มาไว้ที่รีจิสเตอร์ A
MOV LED, A	แสดงผลรูปแบบแรก
CALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
LOOP: INC R2	เพิ่มค่าให้ตัวชี้ข้อมูล R2 ขึ้นอีก 1
MOV A, R2	นำตัวชี้ข้อมูล R2 มาไว้ที่รีจิสเตอร์ A
MOVC A, @A+DPTR	นำค่าในตาราง DATA มาไว้ที่รีจิสเตอร์ A
MOV LED, A	แสดงผลรูปแบบถัดไป
CALL DELAY	เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
CINE R2, #06H, LOOP	ตรวจสอบว่าแสดงผลครบทุกรูปแบบ หรือไม่
SJMP START	หากครบให้กลับไปเริ่มต้นทำงานใหม่
DELAY: MOV R0, #0FH	โปรแกรมย่อยหน่วงเวลา
DELAY1: MOV R1, #0FFH	โปรแกรมย่อยหน่วงเวลา 1
DELAY2: DJNZ R1, DELAY2	โปรแกรมย่อยหน่วงเวลา 2
DJNZ R0, DELAY1	
RET	สิ้นสุดโปรแกรมหน่วงเวลา
DATA: DB 81H, 0C3H, 0E7H, 0FFH, 0E7H, 0C3H, 81H	ตาราง DATA เก็บข้อมูลแบบการแสดงผล
END	จบการทำงาน