ปฏิบัติการทดลองที่ 4

เสนอ

ผู้ช่วยศาตราจารย์ดร.พนัส นัถฤทธิ์

จัดทำโดย

58364876 นายอาทิตย์ แซ่ว่าง

58366450 นายศิวศิษฎ์ สารขาว

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา 305281 ไมโครโพรเซสเซอร์และภาษาแอสเซมบลี

ภาคเรียนที่ 1/ปีการศึกษา 2560

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สาขาวิชาวิศกรรมคอมพิวเตอร์

มหาวิทยาลัยนเรศวร

คำนำ

รายงาน ปฏิบัติการทดลองที่ 4 ซึ่งเป็นส่วนหนึ่งของรายวิชา 305381 ไมโครโพรเซสเซอร์และภาษาแอสเซมบลี (Microprocessor and Assembly Language) จัดทำขึ้นเพื่อศึกษาการเขียนโปรแกรมภาษาแอสแซมบลีเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 ผ่าน LED

คณะผู้จัดทำหวังว่ารายงานเล่มนี้จะสามารถเป็นประโยชน์ต่อผู้ที่ต้องการศึกษาค้นคว้าข้อมูล 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
5. เพื่อศึกษาการเขียนโปรแกรม คำสั่ง JB, JNB
6. เพื่อศึกษาการเขียนโปรแกรม คำสั่ง SETB, CLR, PUSH, POP และ PWM

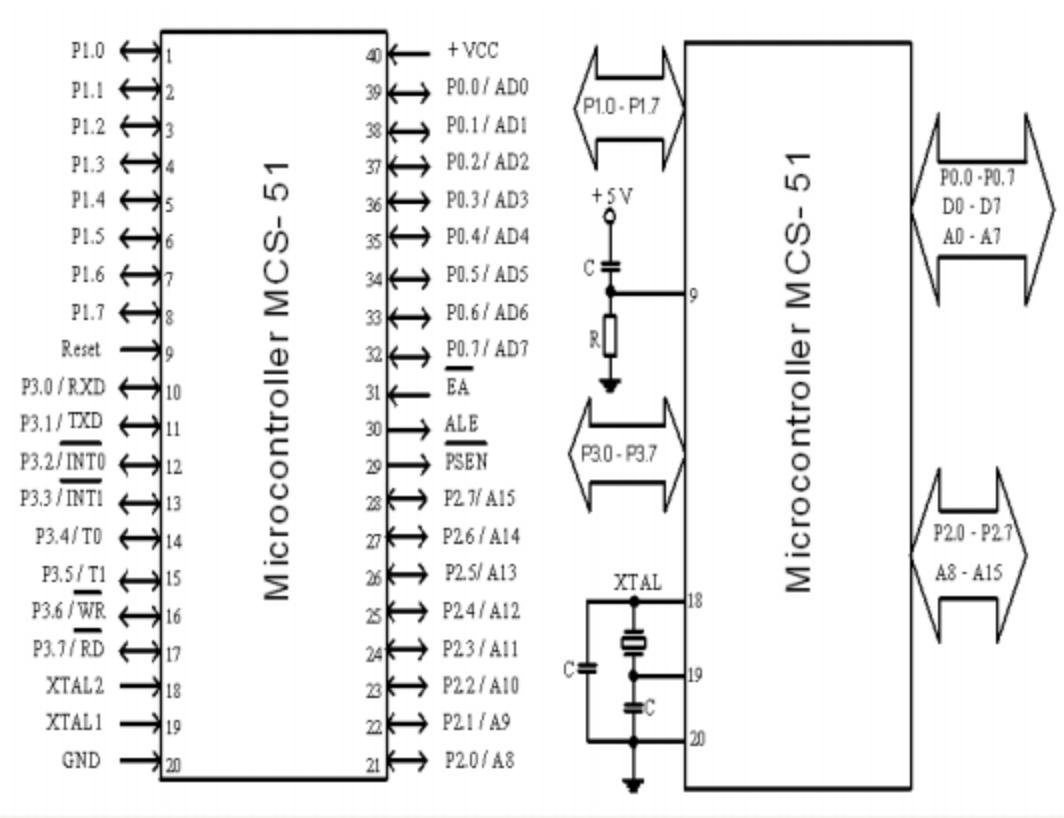
เนื้อหาและทฤษฎี

<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



ไมโครคอนโทรลเลอร์ MCS-51

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

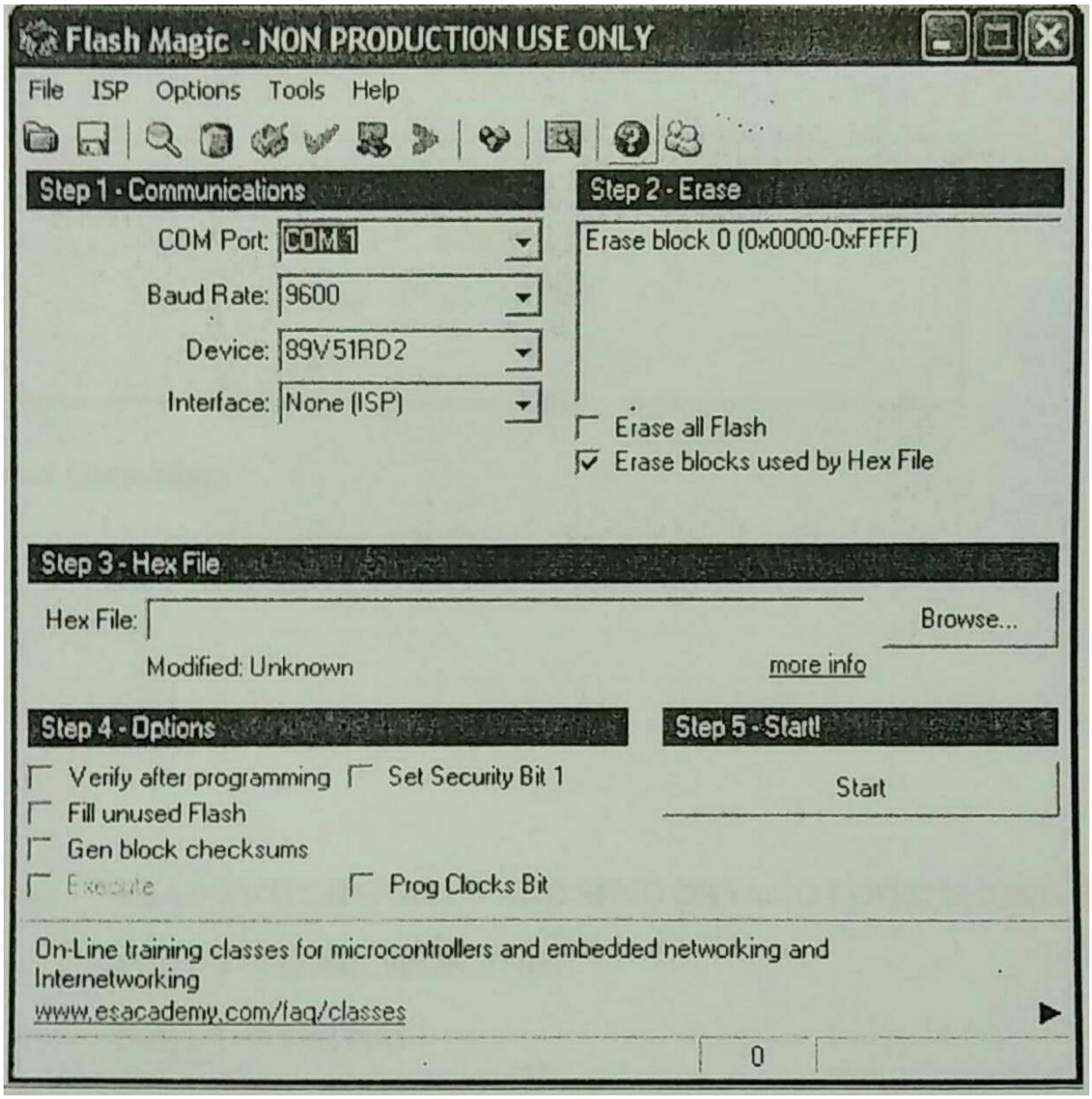
<u>การใช้งานโปรแกรม Flash Magic สำหรับดาวน์โหลดโปรแกรมลงบอร์ดไมโครคอนโทรลเลอร์</u>

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

การทดลองในรายวิชานี้มีวัตถุประสงค์เพื่อเรียนรู้วิธีการเขียนโปรแกรมภาษาแอสแซมบลีสำหรับใช้ควบคุมการทำงานของไมโครคอนโทรเลอร์ตระกูล MCS-51 โดยเนื้อหาในการทดลองจะประกอบด้วยการทดลองจำนวน 5 การทดลอง ดังนี้

ชุดคำสั่งทางลอจิก

ANL A , #data ; ทำการแอนด์ค่าของข้อมูลในแอกคิวมูเลเตอร์ กับข้อมูล data ขนาด 8 บิต แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

ANL A , direct ; ทำการแอนด์ค่าของข้อมูลในหน่วยความจำข้อมูลภายในกับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ANL A , Rn ; ทำการแอนด์ค่าของข้อมูลในรีจีสเตอร์ R0-R7 กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ANL A , @Rn ; ทำการแอนด์ค่าของข้อมูลในหน่วยความจำข้อมูลภายในที่ถูกชี้ โดยรีจีสเตอร์ R0 หรือ R1 กับค่าในรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ANL direct , A ; ทำการแอนด์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายในกับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ANL direct , #data ; ทำการแอนด์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายใน กับข้อมูล data ขนาด 8 บิต แล้วนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ANL C , bit ; ทำการแอนด์ค่าของข้อมูลในแฟลกทด กับค่าของข้อมูลในระดับบิตของรีจีสเตอร์ แล้วนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

ANL C , bit\ ; ทำการแอนด์ค่าของข้อมูลในแฟลกทด กับค่าคอมพลีเมนต์ ของข้อมูลในระดับบิต ของรีจีสเตอร์ โดยข้อมูลของรีจีสเตอร์ไม่มีการเปลี่ยนแปลง จากนั้นนำผลลัพธ์ไปเก็บไว้ใน แฟลกทด (ค่าคอมพลีเมนต์ คือค่าที่ตรงข้ามกับค่าของข้อมูล)

ORL A , #data ; ทำการออร์ค่าในแอกคิวมูเลเตอร์ กับข้อมูล data ขนาด 8 บิต นำผลลัพธ์ ไปเก็บไว้ในแอกคิวมูเลเตอร์

ORL A , direct ; ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ORL A , Rn ; ทำการออร์ค่าของข้อมูลในรีจีสเตอร์ R0-R7 กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ORL A , @Rn ; ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน ที่ถูกชี้โดยรีจีสเตอร์ R0 หรือ R1 กับค่าในรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

ORL direct , A ; ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ORL A,#data ; ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายใน กับข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

ORL C , bit ; ทำการออร์ค่าของข้อมูลในแฟลกทด กับค่าของข้อมูลในระดับบิตของรีจีสเตอร์ แล้วนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

ORL C , bit\ ; ทำการออร์ค่าของข้อมูลในแฟลกทด กับค่าคอมพลีเมนต์ของข้อมูลในระดับบิต ของรีจีสเตอร์ โดยข้อมูลของรีจีสเตอร์ไม่มีการเปลี่ยนแปลง จากนั้นนำผลลัพธ์ไปเก็บไว้ในแฟลกทด

XRL A , #data ; ทำการเอ็กคลูซีฟ-ออร์ค่าในแอกคิวมูเลเตอร์กับข้อมูล data ขนาด 8บิต นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

XRL A , direct ; ทำการเอ็กคลูซีฟ-ออร์ค่าในหน่วยความจำข้อมูลภายใน กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

XRL A , Rn ; ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูลในรีจีสเตอร์ R0-R7 กับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

XRL A , @Rn ; ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายใน ที่ถูกชี้โดย รีจีสเตอร์ R0 หรือ R1 กับค่าในรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

XRL direct , A ; ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายใน กับค่าของ รีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

XRL direct , #data ; ทำการเอ็กคลูซีฟ-ออร์ค่าของข้อมูล ในหน่วยความจำข้อมูลภายใน กับข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในหน่วยความจำข้อมูลภายใน

CLR A ; ทำการเคลียร์ค่าของรีจีสเตอร์ A ให้เท่ากับ 00H

CPL A ; ทำการกลับสถานะของข้อมูลในรีจีสเตอร์ A ให้มีค่าตรงข้าม

RL A ; ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางซ้าย บิต 7 จะหมุนวนมายังบิต 0

RLC A ; ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางซ้ายผ่านแฟลกทด โดยบิต 7 จะหมุนไปยังแฟลกทด และข้อมูลของแฟลกทดเดิมจะหมุนเข้ามาในบิต 0

RR A ; ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางขวา บิต 0 จะหมุนวนมายังบิต 7

RRC A ; ทำการหมุนข้อมูลในแต่ละบิตของรีจีสเตอร์ A วนทางขวาผ่านแฟลกทด โดยบิต 0 จะหมุนไปยังแฟลกทด และข้อมูลของแฟลกทดเดิมจะหมุนเข้ามาในบิต 7

SWAP A ; แลกเปลี่ยนข้อมูลระหว่างบิต 0-3 กับบิต 4-7 ของรีจีสเตอร์ A

CLR C ; ทำการเคลียร์ค่าของแฟลกทดให้เท่ากับ "0"

CLR bit ; ทำการเคลียร์ค่าของข้อมูลในบิตที่กำหนดให้เท่ากับ "0"

CPL C ; ทำการคอมพลีเมนต์ หรือกลับสถานะลอจิกของแฟลกทด

CPL bit ; ทำการคอมพลีเมนต์หรือกลับสถานะลอจิกของข้อมูลในบิตที่กำหนด

SETB C ; ทำการเซตค่าของแฟลกทดให้เท่ากับ "1"

SETB bit ; ทำการเซตค่าของข้อมูลในบิตที่กำหนดให้เท่า

MICROCONTROLLER MCS-51 – LAB 4

Summary: BASIC DC MOTOR CONTROL

LAB 4-1

Description: CONTROL THE MOTOR IN FORWARD AND REVERSE DIRECTION

Hardware: PORT 2 -> DC MOTOR

- P2.0 -> IN1
- P2.1 -> IN2
- P2.2 -> EN

NOTE: 1. ให้นิสิตต่อ Vcc และ GND บอร์ด ANALAB เข้ากับ Vcc และ GND ของ Motor และต่อ GND ของบอร์ด ANALAB และ Motor เข้ากับ GND ของ MCS-51

2. ให้นิสิตลองจับสัญญาณที่ OPA และ OPB โดยทำการเชื่อมต่อขาสัญญาณดังกล่าวกับหลอด LED บนบอร์อด ANALAB (Logic Monitor)

ASM Code:

```
                ORG     0000H
                DC_MOTOR     EQU     P2
                MOTOR_PLUS  EQU     DC_MOTOR.0
                MOTOR_MINUS         EQU     DC_MOTOR.1
                MOTOR_EN     EQU     DC_MOTOR.2
START:          MOV     DC_MOTOR, #00H
CCW:            SETB    MOTOR_PLUS
                CLR     MOTOR_MINUS
                SETB    MOTOR_EN
                ACALL   DELAY
                CLR     MOTOR_EN
                ACALL   DELAY
CW:             CLR     MOTOR_PLUS
                SETB    MOTOR_MINUS
                SETB    MOTOR_EN
                ACALL   DELAY
                CLR     MOTOR_EN
                ACALL   DELAY
                SJMP    START

DELAY:          MOV     R0, #5
DELAY0:         MOV     R1, #255
DELAY1:         MOV     R2, #255
DELAY2:         DJNZ    R2, DELAY2
                DJNZ    R1, DELAY1
                DJNZ    R0, DELAY0
                RET
                END
```

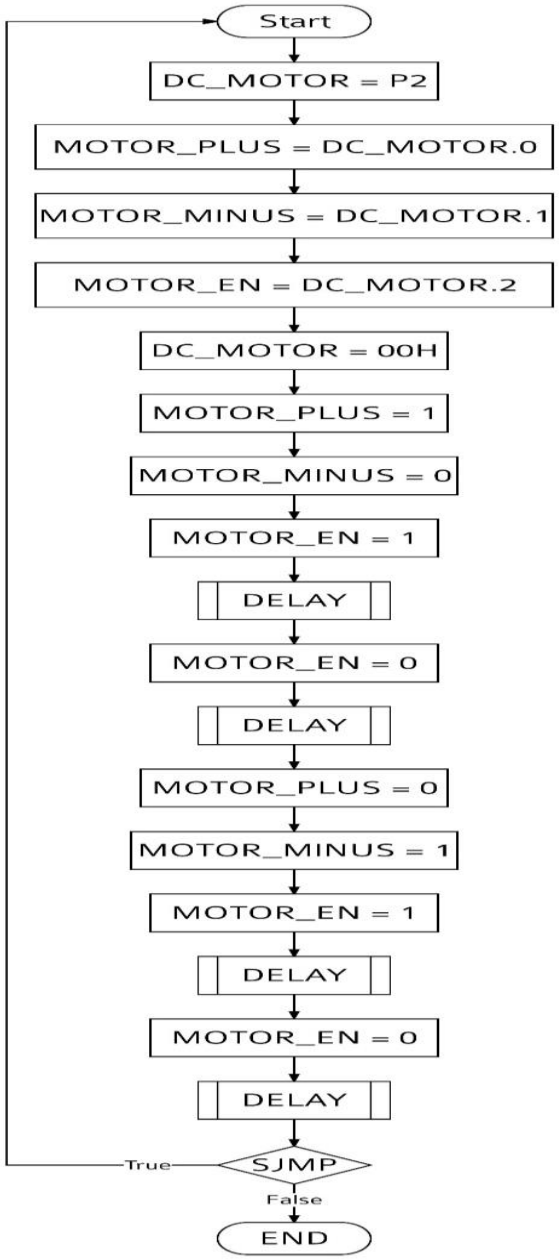
วิธีการทดลอง

1. เขียนโค้ดที่โจทย์กำหนดให้ จากนั้นทำการแปลงไฟล์ให้ได้ไฟล์นามสกุล .hex

2. ทำการรัน ASM Code จากโปรแกรม Flash Magic เพื่อดาวน์โหลดโค้ดลงบอร์ดไมโครคอนโทรลเลอร์ผ่านสายสื่อข้อมูล

3. สังเกตผลการทดลองและบันทึกผลการทดลอง

Flow Cart:



ORG	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
DC_MOTOR EQU P2	ประกาศตัวแปร DC_MOTOR สำหรับเรียกแทนพอร์ต 2
MOTOR_PLUS EQU DC_MOTOR.0	ประกาศตัวแปร MOTOR_PLUS สำหรับเรียกแทนพอร์ต 2 บิตที่ 0
MOTOR_MINUS EQU DC_MOTOR.1	ประกาศตัวแปร MOTOR_MINUS สำหรับเรียกแทนพอร์ต 2 บิตที่ 1
MOTOR_EN EQU DC_MOTOR.2	ประกาศตัวแปร MOTOR_EN สำหรับเรียกแทนพอร์ต 2 บิตที่ 2
START: MOV DC_MOTOR, #00H	เริ่มโปรแกรมย่อย ชื่อ start กำหนดให้ DC_MOTOR = 0 เพื่อใช้เป็นตัวชี้ข้อมูล
CCW: SETB MOTOR_PLUS	เริ่มโปรแกรมย่อย ชื่อ CCW กำหนดให้ค่าใน MOTOR_PLUS เป็น 1
CLR MOTOR_MINUS	ลบค่าใน MOTOR_MINUS เป็น 0
SETB MOTOR_EN	กำหนดให้ค่าใน MOTOR_EN เป็น 1
ACALL DELAY	เริ่มใช้โปรแกรมย่อยเพื่อหน่วงเวลา
CLR MOTOR_EN	ลบค่าใน MOTOR_EN เป็น 0
ACALL DELAY	เริ่มใช้โปรแกรมย่อยเพื่อหน่วงเวลา
CW: CLR MOTOR_PLUS	เริ่มโปรแกรมย่อย ชื่อ CW ลบค่าใน MOTOR_PLUS ให้เป็น 0
SETB MOTOR_MINUS	กำหนดให้ค่าใน MOTOR_PLUS เป็น 1
SETB MOTOR_EN	กำหนดให้ค่าใน MOTOR_EN เป็น 1
ACALL DELAY	เริ่มใช้โปรแกรมย่อยเพื่อหน่วงเวลา
CLR MOTOR_EN	ลบค่าใน MOTOR_EN เป็น 0
ACALL DELAY	เริ่มใช้โปรแกรมย่อยเพื่อหน่วงเวลา
SJMP START	กระโดดไปยังคำสั่ง START
DELAY: MOV R0, #5	เริ่มโปรแกรมย่อย ชื่อ DELAY ย้ายค่า 5 ไปเป็นข้อมูลในรีจิสเตอร์ R0
DELAY0: MOV R1, #255	เริ่มโปรแกรมย่อย ชื่อ DELAY0 ย้ายค่า 255 ไปเป็นข้อมูลในรีจิสเตอร์ R1
DELAY1: MOV R2, #255	เริ่มโปรแกรมย่อย ชื่อ DELAY1 ย้ายค่า 5 ไปเป็นข้อมูลในรีจิสเตอร์ R2
DELAY2: DJNZ R2, DELAY2	เริ่มโปรแกรมย่อย ชื่อ DELAY2 ลดค่าใน R2 แล้วไปยังคำสั่ง DELAY2
DJNZ R1, DELAY1	ลดค่าใน R1 แล้วไปยังคำสั่ง DELAY1
DJNZ R0, DELAY0	ลดค่าใน R0 แล้วไปยังคำสั่ง DELAY0

RET	สิ้นสุดโปรแกรมหน่วงเวลา
END	สิ้นสุดโปรแกรม

Result and discussion:

มอเตอร์หมุนทวนเข็มนาฬิกา หยุดเป็นเวลา 3 วินาที แล้วหมุนตามเข็มนาฬิกา หยุดเป็นเวลา 3 วินาที แล้กลับไปหมุนทวนเข็มนาฬิกาเป็นแบบนี้ไปเรื่อยๆ

LAB 4-2

Description: CONTROL THE MOTOR IN FORWARD/REVERSE/STOP DIRECTION BY SWITCH

Hardware: PORT 1 -> SWITCH

- P1.0 -> SW0
- P1.1 -> SW1

PORT 2 -> DC_MOTOR

- P2.0 -> IN1
- P2.1 -> IN2
- P2.2 -> EN

ASM Code:

```
                ORG     0000H
                DC_MOTOR     EQU     P2
                MOTOR_PLUS   EQU     DC_MOTOR.0
                MOTOR_MINUS          EQU     DC_MOTOR.1
                MOTOR_EN     EQU     DC_MOTOR.2
                SWITCH       EQU     P1
                SW0          EQU     SWITCH.0
                SW1          EQU     SWITCH.1
START:          MOV     DC_MOTOR, #00H
                MOV     SWITCH, #0FFH
CHK:            JB      SW0, CHK_SW1
                JB      SW1, CW
                CLR     MOTOR_EN
                SJMP    CHK
CHK_SW1         JB      SW1, CHK
CCW:            SETB    MOTOR_PLUS
                CLR     MOTOR_MINUS
                SJMP    CHK
CW:             CLR     MOTOR_PLUS
                SETB    MOTOR_MINUS
                SETB    MOTOR_EN
                SJMP    CHK
                END
```

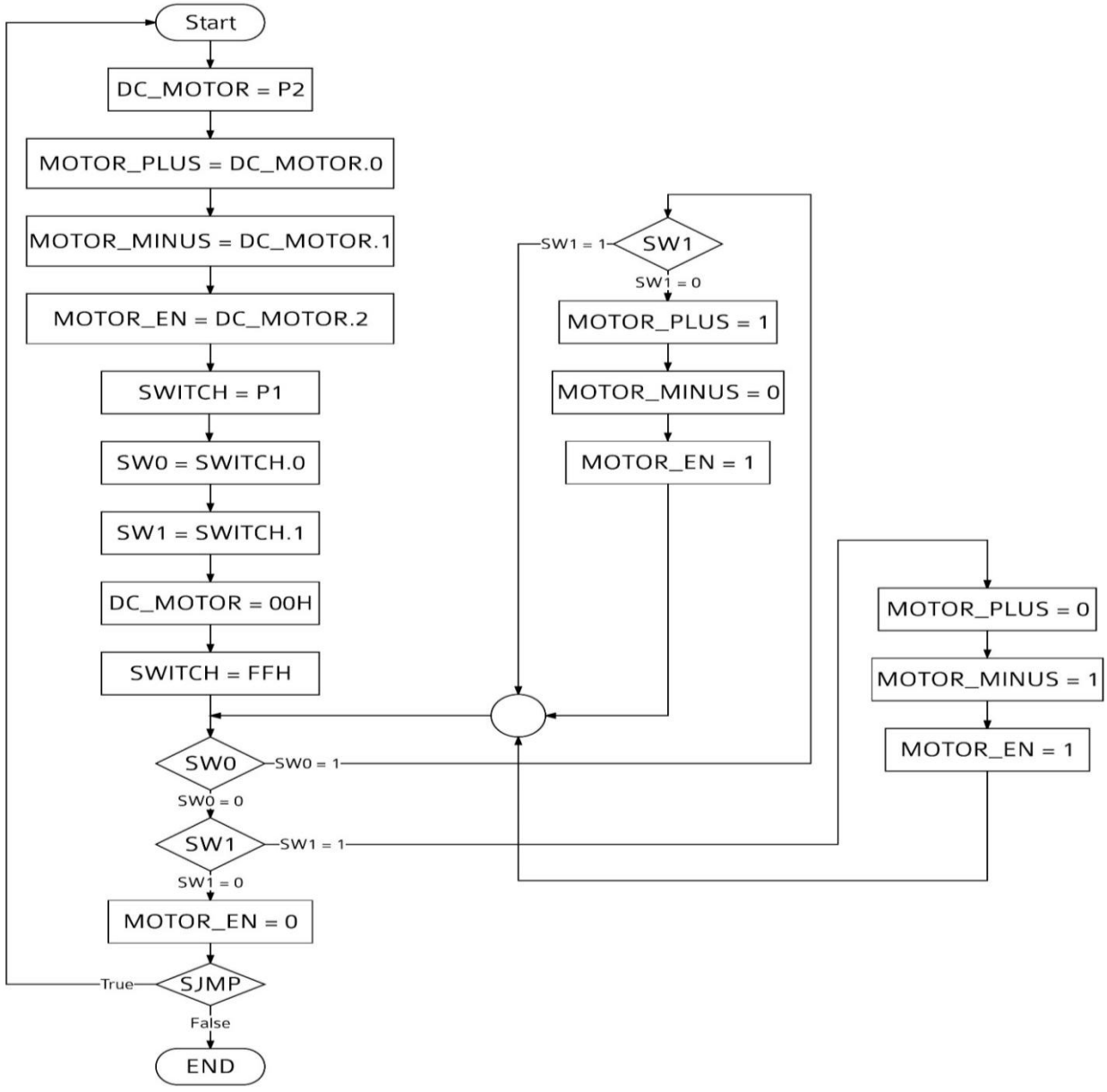
วิธีการทดลอง

1. เขียนโค้ดที่โจทย์กำหนดให้ จากนั้นทำการแปลงไฟล์ให้ได้ไฟล์นามสกุล .hex

2. ทำการรัน ASM Code จากโปรแกรม Flash Magic เพื่อดาวน์โหลดโค้ดลงบอร์ดไมโครคอนโทรลเลอร์ผ่านสายสื่อข้อมูล

3. สังเกตผลการทดลองและบันทึกผลการทดลอง

Flow Cart:



ORG	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
DC_MOTOR EQU P2	ประกาศตัวแปร DC_MOTOR สำหรับเรียกแทนพอร์ต 2
MOTOR_PLUS EQUDC_MOTOR.0	ประกาศตัวแปร MOTOR_PLUS สำหรับเรียกแทนพอร์ต 2 บิตที่ 0
MOTOR_MINUS EQUDC_MOTOR.1	ประกาศตัวแปร MOTOR_MINUS สำหรับเรียกแทนพอร์ต 2 บิตที่ 1
MOTOR_EN EQU DC_MOTOR.2	ประกาศตัวแปร MOTOR_EN สำหรับเรียกแทนพอร์ต 2 บิตที่ 2
SWITCH EQU P1	ประกาศตัวแปร SWITCH สำหรับเรียกแทนพอร์ต 1
SW0 EQU SWITCH.0	ประกาศตัวแปร SW0 สำหรับเรียกแทนพอร์ต 1 บิตที่ 0
SW1 EQU SWITCH.1	ประกาศตัวแปร SW1 สำหรับเรียกแทนพอร์ต 1 บิตที่ 1
START: MOV DC_MOTOR, #00H	เริ่มโปรแกรมย่อย ชื่อ START กำหนดให้ DC_MOTOR = 0 เพื่อใช้เป็นตัวชี้ข้อมูล
MOV SWITCH, #0FFH	กำหนดให้ SWITCH = FFH เพื่อใช้เป็นตัวชี้ข้อมูล
CHK: JB SW0, CHK_SW1	เริ่มโปรแกรมย่อย ชื่อ CHK ถ้า SW0 ถูกกด ให้ไปที่โปรแกรม CHK_SW1
JB SW1, CW	ถ้า SW1 ถูกกด ให้ไปที่โปรแกรม CW
CLR MOTOR_EN	กำหนดให้ค่าใน MOTOR_EN เป็น 0
SJMP CHK	ไปที่คำสั่ง CHK
CCW: SETB MOTOR_PLUS	เริ่มโปรแกรมย่อย ชื่อ CCW กำหนดให้ค่าใน MOTOR_PLUS เป็น 1
CLR MOTOR_MINUS	กำหนดให้ค่าใน MOTOR_MINUS เป็น 0
SJMP CHK	ไปที่คำสั่ง CHK
CW: CLR MOTOR_PLUS	เริ่มโปรแกรมย่อย ชื่อ CW กำหนดให้ค่าใน MOTOR_PLUS เป็น 0
SETB MOTOR_MINUS	กำหนดให้ค่าใน MOTOR_MINUS เป็น 1
SETB MOTOR_EN	กำหนดให้ค่าใน MOTOR_EN เป็น 1
SJMP CHK	ไปที่คำสั่ง CHK
END	สิ้นสุดโปรแกรม

Results and discussion:

ถ้า SW0 และ SW1 ไม่ถูกกดทั้งคู่ มอเตอร์จะหยุด ถ้า SW0 ถูกกดแต่ SW1 ไม่ถูกกด มอเตอร์จะหมุนทวนเข็มนาฬิกา ถ้า SW0 ไม่ถูกกดแต่ SW1 ถูกกด มอเตอร์จะหมุนตามเข็มนาฬิกา ถ้า SW0 และ SW1 ถูกกดทั้งคู่ มอเตอร์จะทำตามสถานะก่อนหน้าที่ สวิตช์ทั้งสองจะถูกกด

LAB 4-3

Description: SPEED CONTROL FOR DC MOTOR BY PWM METHOD

Hardware: PORT 1 -> SWITCH

- P1.0 -> SW0

PORT 2 -> DC MOTOR

- P2.0 -> IN1
- P2.1 -> IN2
- P2.2 -> EN

NOTE: ให้นิสิตสังเกตการณ์ทำงานและการเปลี่ยนแปลงของมอเตอร์

ASM Code:

```
                ORG     0000H
                DC_MOTOR    EQU     P2
                MOTOR_PLUS  EQU     DC_MOTOR.0
                MOTOR_MINUS         EQU     DC_MOTOR.1
                MOTOR_EN    EQU     DC_MOTOR.2
                SWITCH      EQU     P1
                SW0         EQU     SWITCH.0

START:          MOV     SWITCH, #0FFH
                MOV     DC_MOTOR, #00H
                CLR     MOTOR_EN
CCW:            SETB    MOTOR_PLUS
                CLR     MOTOR_MINUS

MAIN_LOOP:      JB      SW0, HI_SPEED
                MOV     R0, #40
                MOV     R1, #60
                LCALL   PWM_CONTROL
                SJMP    MAIN_LOOP

HI_SPEED:       MOV     R0, #180
                MOV     R1, #10
                LCALL   PWM_CONTROL
                SJMP    MAIN_LOOP

PWM_CONTROL:        PUSH    00H
                    PUSH    01H
                    SETB    MOTOR_EN
PWM_ON_LOOP:        LCALL   PWM_DELAY
                    DJNZ    R0, PWM_ON_LOOP
                    CLR     MOTOR_EN
```

```
PWM_OFF_LOOP:       LCALL  PWM_DELAY
                    DJNZ   R1, PWM_OFF_LOOP
                    POP    01H
                    POP    00H
                    RET
PWM_DELAY:  PUSH    07H
            MOV     R7, #100
PWM_DELAY1  NOP
            DJNZ    R7, PWM_ DELAY1
            POP     07H
            RET
            END
```

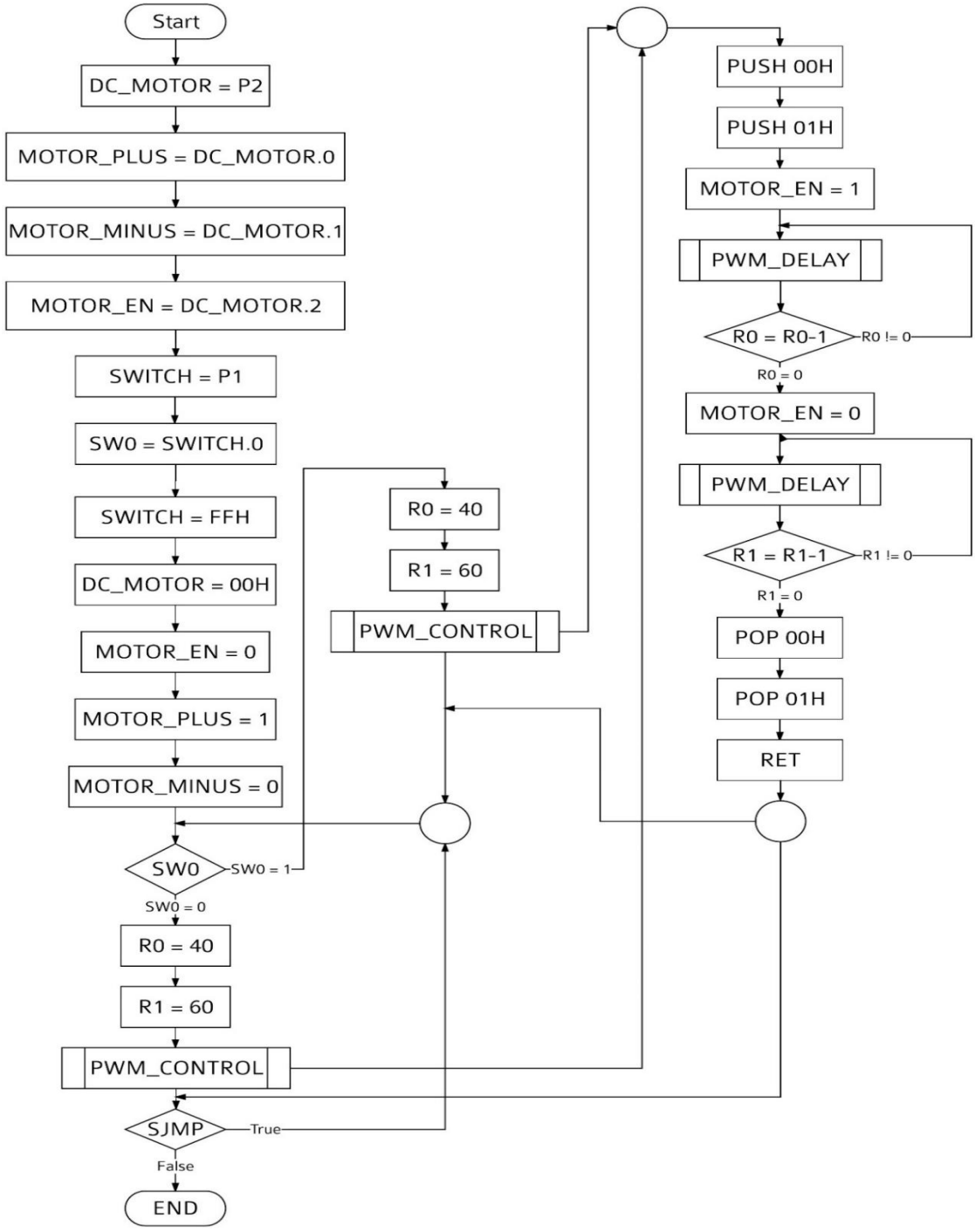
วิธีการทดลอง

1. เขียนโค้ดที่โจทย์กำหนดให้ จากนั้นทำการแปลงไฟล์ให้ได้ไฟล์นามสกุล .hex

2. ทำการรัน ASM Code จากโปรแกรม Flash Magic เพื่อดาวน์โหลดโค้ดลงบอร์ดไมโครคอนโทรลเลอร์ผ่านสายสื่อข้อมูล

3. สังเกตผลการทดลองและบันทึกผลการทดลอง

Flow Cart:


Start
DC_MOTOR = P2
MOTOR_PLUS = DC_MOTOR.0
MOTOR_MINUS = DC_MOTOR.1
MOTOR_EN = DC_MOTOR.2
SWITCH = P1
SW0 = SWITCH.0
SWITCH = FFH
DC_MOTOR = 00H
MOTOR_EN = 0
MOTOR_PLUS = 1
MOTOR_MINUS = 0
SW0
SW0 = 1
SW0 = 0
R0 = 40
R1 = 60
PWM_CONTROL
SJMP
True
False
END
R0 = 40
R1 = 60
PWM_CONTROL
PUSH 00H
PUSH 01H
MOTOR_EN = 1
PWM_DELAY
R0 = R0-1
R0 != 0
R0 = 0
MOTOR_EN = 0
PWM_DELAY
R1 = R1-1
R1 != 0
R1 = 0
POP 00H
POP 01H
RET

ORG	เริ่มต้นการทำงานที่ตำแหน่ง 0000H
DC_MOTOR EQU P2	ประกาศตัวแปร DC_MOTOR สำหรับเรียกแทนพอร์ต 2
MOTOR_PLUS EQUDC_MOTOR.0	ประกาศตัวแปร MOTOR_PLUS สำหรับเรียกแทนพอร์ต 2 บิตที่ 0
MOTOR_MINUS EQUDC_MOTOR.1	ประกาศตัวแปร MOTOR_MINUS สำหรับเรียกแทนพอร์ต 2 บิตที่ 1
MOTOR_EN EQU DC_MOTOR.2	ประกาศตัวแปร MOTOR_EN สำหรับเรียกแทนพอร์ต 2 บิตที่ 2
SWITCH EQU P1	ประกาศตัวแปร SWITCH สำหรับเรียกแทนพอร์ต 1
SW0 EQU SWITCH.0	ประกาศตัวแปร SW0 สำหรับเรียกแทนพอร์ต 1 บิตที่ 0
START: MOV DC_MOTOR, #0FFH	เริ่มโปรแกรมย่อย ชื่อ start กำหนดให้ DC_MOTOR = FFH เพื่อใช้เป็นตัวชี้ข้อมูล
MOV SWITCH, #00H	กำหนดให้ SWITCH = 0 เพื่อใช้เป็นตัวชี้ข้อมูล
CCW: SETB MOTOR_PLUS	เริ่มโปรแกรมย่อย ชื่อ CCW กำหนดให้ค่าใน MOTOR_PLUS เป็น 1
CLR MOTOR_MINUS	ลบค่าใน MOTOR_MINUS เป็น 0
MAIN_LOOP: JB SW0, HI_SPEED	เริ่มโปรแกรมย่อย ชื่อ MAIN_LOOP ถ้า SW0 ถูกกด ให้ที่คำสั่ง HI_SPEED
MOV R0, #40	ย้าย 40 ไปเป็นข้อมูลใน R0
MOV R1, #60	ย้าย 60 ไปเป็นข้อมูลใน R1
LCALL PWM_CONTROL	เรียกใช้โปรแกรม ชื่อ PWM_CONTROL
SJMP MAIN_LOOP	ไปที่โปรแกรมย่อย MAIN_LOOP
HI_SPEED: MOV R0, #180	เรียกใช้โปรแกรม ชื่อ HI_SPEED ย้าย 180 ไปเป็นข้อมูลใน R0
MOV R1, #10	ย้าย 10 ไปเป็นข้อมูลใน R1
LCALL PWM_CONTROL	เรียกใช้โปรแกรม ชื่อ PWM_CONTROL
SJMP MAIN_LOOP	ไปที่โปรแกรมย่อย MAIN_LOOP
PWM_CONTROL: PUSH 00H	เรียกใช้โปรแกรม ชื่อ PWM_CONTROLเก็บค่าตำแหน่งก่อนหน้าไว้ที่ 00
PUSH 01H	เก็บค่าตำแหน่งก่อนหน้าไว้ที่ 01
SETB MOTOR_EN	กำหนดให้ MOTOR_EN เป็น 1
PWM_ON_LOOOP: LCALL PWM_DELAY	เรียกใช้โปรแกรม ชื่อ PWM_ON_LOOOP เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
DJNZ R0, PWM_ON_LOOP	ลดค่าใน R0 ลง 1 แล้วไปที่คำสั่ง PWM_ON_LOOP
CLR MOTOR_EN	กำหนดให้ MOTOR_EN เป็น 0

PWM_OFF_LOOOP: LCALL PWM_DELAY	เรียกใช้โปรแกรม ชื่อ PWM_OFF_LOOOP เรียกใช้โปรแกรมย่อยเพื่อหน่วงเวลา
DJNZ R0, PWM_OFF_LOOP	ลดค่าใน R0 ลง 1 แล้วไปที่คำสั่ง PWM_OFF_LOOP
POP 01H	ดึงค่าที่เก็บไว้ใน 01
POP 00H	ดึงค่าที่เก็บไว้ใน 00
RET	สิ้นสุด
PWM_DELAY: PUSH 07H	เริ่มโปรแกรมย่อย ชื่อ PWM_DELAY เก็บค่าไว้ใน 07
MOV R7, #100	ย้าย 100 ไปเป็นข้อมูลใน R7
PWM_DELAY1: NOP	เริ่มโปรแกรมย่อย ชื่อ PWM_DELAY1
DJNZ R7, PWM_DELAY1	ลดค่าใน R7 ลง 1 แล้วไปที่คำสั่ง PWM_DELAY1
POP 07H	ดึงค่าใน 07
RET	สิ้นสุดโปรแกรมหน่วงเวลา
END	สิ้นสุดโปรแกรม

Results and discussion:

มีการกำหนดความเร็วของมอเตอร์ และสั่งให้มอเตอร์ทำงานจากการเก็บค่าแล้วดึงค่าที่เก็บออกมาควบคุมมอเตอร์

EXERCISE

1. จงดัดแปลงโปรแกรม LAB 4-3 โดยเพิ่มการใช้งาน SW0 -SW3 ให้มีการทำงานดังนี้

- หากไม่มีการสับสวิตช์ (SW0 - SW3 = 0) ให้มอเตอร์หมุนต่ำที่สุด
- หากสับสวิตช์ SW0 (SW1 - SW3 = 0) ให้มอเตอร์หมุนที่ความเร็วปกติ
- หากสับสวิตช์ SW1 (SW0, SW2, SW3 = 0) ให้มอเตอร์หมุนที่ความเร็วสูงขึ้น
- หากสับสวิตช์ SW2 (SW0, SW1, SW3 = 0) ให้มอเตอร์หมุนที่ความเร็วสูงมาก
- หากสับสวิตช์ SW3 (SW0, SW1, SW2 = 0) ให้มอเตอร์หมุนที่ความเร็วสูงที่สุด

ตอบ

```
        ORG 0000H
DC_MOTOR    EQU P2
MOTOR_PLUS  EQU DC_MOTOR.0
MOTOR_MINUS     EQU DC_MOTOR.1
MOTOR_EN    EQU DC_MOTOR.2
SWITCH          EQU P1
SW0         EQU SWITCH.0
SW1         EQU SWITCH.1
SW2         EQU SWITCH.2
SW3         EQU SWITCH.3

START:  MOV SWITCH, #0FFH
        MOV DC_MOTOR, #00H
        CLR MOTOR_EN

CCW:    SETB MOTOR_PLUS
        CLR MOTOR_MINUS

MAIN_LOOP:  JB SW0, NORMAL1
        JB SW1, HIGH1
        JB SW2, HIGHER1
        JB SW3, HIGHEST1
```

```
                MOV R0, #10
                MOV R1, #90
                LCALL PWM_CONTROL
                SJMP MAIN_LOOP

NORMAL1:        JB SW1, MAIN_LOOP
                JB SW2, MAIN_LOOP
                JB SW3, MAIN_LOOP
                MOV R0, #30
                MOV R1, #70
                LCALL PWM_CONTROL
                SJMP MAIN_LOOP

HIGH1:          JB SW2, MAIN_LOOP
                JB SW3, MAIN_LOOP
                MOV R0, #50
                MOV R1, #50
                LCALL PWM_CONTROL
                SJMP MAIN_LOOP

HIGHER1:        JB SW3, MAIN_LOOP
                MOV R0, #70
                MOV R1, #30
                LCALL PWM_CONTROL
                SJMP MAIN_LOOP

HIGHEST1:       MOV R0, #90
                MOV R1, #10
                LCALL PWM_CONTROL
                SJMP MAIN_LOOP

PWM_CONTROL:    PUSH 00H
                PUSH 01H
```

```
                SETB MOTOR_EN

PWM_ON_LOOP:    LCALL PWM_DELAY
                DJNZ R0, PWM_ON_LOOP
                CLR MOTOR_EN
PWM_OFF_LOOP:   LCALL PWM_DELAY
                DJNZ R1, PWM_OFF_LOOP
                POP 01H
                POP 00H
                RET

PWM_DELAY:      PUSH 07H
                MOV R7, #100

PWM_DELAY1: NOP
                DJNZ R7, PWM_DELAY1
                POP 07H
                RET
                END
```

วิเคราะห์ผลการทดลอง

เมื่อเราใช้สวิตช์ 2 ตัวในการจ่ายไฟให้กับมอเตอร์ ขณะที่ไม่จ่ายไฟทั้ง 2 สวิตช์ มอเตอร์จะไม่หมุนแต่เมื่อจ่ายไฟ สวิตช์ตัวที่ 1 แต่สวิตช์ตัวที่สองไม่จ่าย มอเตอร์จะหมุนทวนเข็มนาฬิกา และเมื่อสวิตช์ตัวที่ 2 จ่ายไฟแต่สวิตช์ตัวที่ 1 ไม่ได้จ่ายไฟ มอเตอร์จะหมุนตามเข็มนาฬิกา เมื่อจ่ายไฟทั้ง 2 สวิตช์มอเตอร์จะทำงานเหมือนก่อนที่สวิตช์ทั้ง 2 จะจ่ายไฟพร้อมกัน

สามารถเพิ่มความเร็วให้กับมอเตอร์จากการเขียนโค้ด

สรุปผลการทดลอง

มอเตอร์สามารถหมุนได้ 2 ทิศทางคือ ตามเข็มนาฬิกากับทวนเข็มนาฬิกา และสามารถปรับความเร็วได้

เอกสารอ้างอิง

http://www.mind-tek.net/8051.php

http://www.ett.co.th/article/Robot/et_robot_rd2/rd2_001.html

http://www.olearning.siam.edu/2011-11-28-08-10-01/593-100-101-

http://www.thaiall.com/assembly/asmcommand.htm

http://mcs51-yongyoot.blogspot.com/2008/12/blog-post_07.html