

	ใบเนื้อหา	หน้าที่ 1
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์
<div>หน่วยที่ 1 พื้นฐานของวงจรดิจิทัลและไมโครคอนโทรลเลอร์</div> <div>พื้นฐานไมโครคอนโทรลเลอร์</div> <div>1. ความหมายและคุณสมบัติของไมโครคอนโทรลเลอร์</div> <div>1.1 ความหมายของไมโครคอนโทรลเลอร์</div> <p>ไมโครคอนโทรลเลอร์ (Microcontroller เรียกว่า μC, uC หรือ MCU) หมายถึงอุปกรณ์ไอซีประเภทการประมวลผล ที่รวมเอาความสามารถที่คล้ายคลึงกับไมโครคอมพิวเตอร์บรรจุเข้าไว้ในตัวถังเดียวกัน แล้วสามารถนำไปต่อใช้งานได้เลย เพราะภายในของไมโครคอนโทรลเลอร์จะประกอบด้วย 5 องค์ประกอบสำคัญ ได้แก่ ส่วนที่เป็นขาสัญญาณที่ใช้ในการติดต่อกับอุปกรณ์อินพุต ,ส่วนที่เป็นขาสัญญาณที่ใช้ในการติดต่อกับอุปกรณ์เอาต์พุต ,ส่วนที่ใช้ในการประมวลผลข้อมูลในรูปแบบคณิตศาสตร์ และลอจิก ที่นิยมเรียกว่า CPU ,ส่วนที่เป็นหน่วยความจำ และส่วนสุดท้ายคือส่วนโมดูลที่ทำหน้าที่พิเศษอื่น ๆ ซึ่งไมโครคอนโทรลเลอร์เป็นอุปกรณ์ที่สามารถเขียนโปรแกรมควบคุมการทำงานไว้ภายในไอซีได้</p> <div></div> <p>รูปที่ 1.1 รูปตัวอย่างของไมโครคอนโทรลเลอร์ตัวถังแบบ SMD (Surface Mount Device) (ที่มา : www.engineer007.com/articles/507518/)</p> <div>1.2 สถาปัตยกรรมของไมโครคอนโทรลเลอร์</div> <p>สถาปัตยกรรมของไมโครคอนโทรลเลอร์ จะกล่าวถึงส่วนสำคัญทั้งหมด 3 ส่วน ได้แก่ โครงสร้างของไมโครคอนโทรลเลอร์ สถาปัตยกรรมของไมโครคอนโทรลเลอร์ที่เกี่ยวข้องกับการเชื่อมต่อหน่วยความจำเพื่อประมวลผล และสถาปัตยกรรมของไมโครคอนโทรลเลอร์ที่กล่าวถึงลักษณะของการประมวลผล</p> <div>1.2.1 โครงสร้างของไมโครคอนโทรลเลอร์</div> <p>ไมโครคอนโทรลเลอร์ ส่วนใหญ่จะมีโครงสร้างภายในที่เหมือนกัน คือภายในของไมโครคอนโทรลเลอร์จะมีลักษณะคล้ายคลึงกับไมโครคอมพิวเตอร์ที่ประกอบด้วย 4 องค์ประกอบสำคัญ ได้แก่ ส่วนที่เป็นขาสัญญาณที่ใช้ในการติดต่อกับอุปกรณ์อินพุตในรูปแบบต่าง ๆ ทั้งในรูปแบบสัญญาณดิจิทัล สัญญาณอนาล็อก ,ส่วนที่เป็นขาสัญญาณที่ใช้ในการติดต่อกับอุปกรณ์เอาต์พุตในรูปแบบต่าง ๆ ทั้งในรูปแบบสัญญาณดิจิทัล สัญญาณอนาล็อก ,ส่วนที่ใช้ในการประมวลผลข้อมูลในรูปแบบทางคณิตศาสตร์ และลอจิก ที่เรานิยมเรียกว่า CPU และส่วนที่เป็นหน่วยความจำข้อมูล แล้วทำการเพิ่มส่วนพิเศษเข้ามาอีกส่วนหนึ่งเพื่อใช้ให้ทำหน้าที่พิเศษอื่น ๆ</p>

	ใบเนื้อหา		หน้าที่ 2
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

โดยโครงสร้างภายในของไมโครคอนโทรลเลอร์ส่วนต่าง ๆ จะถูกเชื่อมต่อกันด้วยระบบบัสเหมือนกับไมโครโพรเซสเซอร์ ได้แก่ Data Bus ,Address Bus และ Control Bus ดังรูปที่ 1.2




รูปที่ 1.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์

- Input คือส่วนของขาสัญญาณที่ใช้ในการติดต่อกับอุปกรณ์อินพุตทั้งในรูปแบบสัญญาณดิจิทัลและอนาล็อก ซึ่งขึ้นอยู่กับข้อกำหนดคุณสมบัติพิเศษของขาสัญญาณที่ทำหน้าที่เป็นขาสัญญาณอินพุต
- Output คือส่วนของขาสัญญาณที่ใช้ในการติดต่อกับอุปกรณ์เอาต์พุตทั้งในรูปแบบสัญญาณดิจิทัลและอนาล็อก ซึ่งขึ้นอยู่กับข้อกำหนดคุณสมบัติพิเศษของขาสัญญาณที่ทำหน้าที่เป็นขาสัญญาณเอาต์พุต
- CPU คือส่วนที่ใช้ในการประมวลผลข้อมูลทางคณิตศาสตร์ และลอจิก ซึ่งจะมีทั้งแบบ 8 บิต ,16 บิต และ 32 บิต โดยขึ้นอยู่กับตระกูลและเบอร์ของไมโครคอนโทรลเลอร์ที่เราเลือกใช้งาน
- Memory คือส่วนที่เป็นหน่วยความจำข้อมูลที่มีทั้งส่วนที่เป็น ROM และ RAM โดยส่วนที่เป็น ROM จะใช้สำหรับเก็บโปรแกรมที่เราเขียนขึ้นมาให้ไมโครคอนโทรลเลอร์ทำงานต่าง ๆ ส่วน RAM จะเป็นส่วนที่ใช้ในการเก็บข้อมูลขณะที่ไมโครคอนโทรลเลอร์ทำการประมวลผลตามโปรแกรม และจะมีการแบ่งพื้นที่บางส่วนของ RAM ทำหน้าที่เป็นหน่วยความจำ Stack สำหรับไมโครคอนโทรลเลอร์บางประเภท ส่วนอีกประเภทหน่วยความจำ Stack จะถูกแยกออกจากหน่วยความจำ RAM ที่ใช้งานทั่วไป
- Special Module คือส่วนพิเศษที่เพิ่มเติมเข้ามาเพื่อทำหน้าที่พิเศษอื่น ๆ เช่น Module I2C ,SPI ,UART ,PWM และ ADC เป็นต้น โดยไมโครคอนโทรลเลอร์แต่ละตระกูล แต่เบอร์จะมีส่วนพิเศษที่เพิ่มเติมเข้ามาไม่เหมือนกัน ขึ้นอยู่กับบริษัทผู้ผลิต

1.2.2 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ที่เกี่ยวข้องกับการเชื่อมต่อหน่วยความจำ

สถาปัตยกรรมของไมโครคอนโทรลเลอร์ถ้าแบ่งตามลักษณะของการเชื่อมต่อหน่วยความจำ หรือการแบ่งพื้นที่หน่วยความจำของไมโครคอนโทรลเลอร์ จะสามารถแบ่งออกได้เป็น 2 รูปแบบ คือ

	ใบเนื้อหา		หน้าที่ 3
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

- สถาปัตยกรรมแบบฟอนนอยมันน์ (Von Neumann Architecture) โดยตั้งชื่อตามศาสตราจารย์จอห์น ฟอนนอยมันน์ (John Von Neumann) ซึ่งเป็นผู้ออกแบบโครงสร้างและการทำงานของเครื่องคอมพิวเตอร์ มีหลักการคือการประมวลผลทั้งหมดจะกระทำที่หน่วยประมวลผลกลาง คำสั่งและข้อมูลจะถูกเก็บที่หน่วยความจำเดียวกัน เพื่อให้การเก็บและการเรียกใช้ข้อมูลเข้าถึงได้ง่าย แต่มีข้อเสียคือเมื่อเรียกใช้ข้อมูลพร้อมกันการทำงานจะช้า เช่น ไมโครคอนโทรลเลอร์ตระกูล MCS-51

- สถาปัตยกรรมแบบฮาร์วาร์ด (Harvard Architecture) เป็นสถาปัตยกรรมที่ออกแบบเพื่อแก้ปัญหาจุดอ่อนของ Von Neumann โดยแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกันรวมทั้งแยกบัสข้อมูลด้วย ดังนั้นขนาดของข้อมูลไม่จำเป็นต้องเท่ากัน เช่น ส่วนของโปรแกรมมีขนาด 16 บิต แต่ส่วนของข้อมูลมีขนาด 8 บิต เป็นต้น ทำให้การเก็บข้อมูลและการเรียกใช้ข้อมูลทำงานได้เร็วขึ้นซึ่งเป็นสถาปัตยกรรมของไมโครคอนโทรลเลอร์ในยุคปัจจุบัน เช่น ไมโครคอนโทรลเลอร์ตระกูล PIC และ AVR

1.2.3 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ที่กล่าวถึงลักษณะของการประมวลผล

สถาปัตยกรรมของไมโครคอนโทรลเลอร์ที่แบ่งแยกตามลักษณะของการประมวลผลสามารถแบ่งได้เป็น 2 กลุ่ม ได้แก่

- สถาปัตยกรรมการประมวลผลแบบ CISC (Complex Instruction Set Computing) เป็นสถาปัตยกรรมการประมวลผลรูปแบบดั้งเดิมที่จะให้ CPU รองรับการประมวลผลชุดคำสั่งที่มีความซับซ้อนในการประมวลผล และใช้เวลาในการประมวลผลของชุดคำสั่งนั้นมากขึ้น โดยระยะเวลาในการประมวลผลคำสั่งแต่ละชุดคำสั่งจะใช้เวลาไม่เท่ากัน บางคำสั่งเพียงจะใช้เวลาเพียง 1 รอบสัญญาณนาฬิกา และบางคำสั่งจะใช้เวลามากกว่า 1 รอบสัญญาณนาฬิกา ขึ้นอยู่กับความซับซ้อนของชุดคำสั่ง รวมถึงกระบวนการถอดรหัสชุดคำสั่งของโปรแกรมต้องทำงานเรียงตามลำดับคำสั่งจนเสร็จจึงจะสามารถไปทำงานชุดคำสั่งต่อไปได้ ซึ่งเป็นข้อด้อยของสถาปัตยกรรมแบบ CISC เพราะจะทำให้การประมวลผลโดยรวมของ CPU ในรูปแบบนี้ช้ากว่ารูปแบบอื่น อย่างไรก็ตามสถาปัตยกรรมแบบ CISC นี้ยังมีการใช้งานอย่างแพร่หลายและได้พัฒนาต่อเนื่องมาจนถึงปัจจุบัน อย่างเช่น CPU ที่ใช้ในไมโครคอนโทรลเลอร์ตระกูล MCS-51, 68HCxx และ Z80-Encore เป็นต้น


Fetch

Execute

1 Machine Cycle

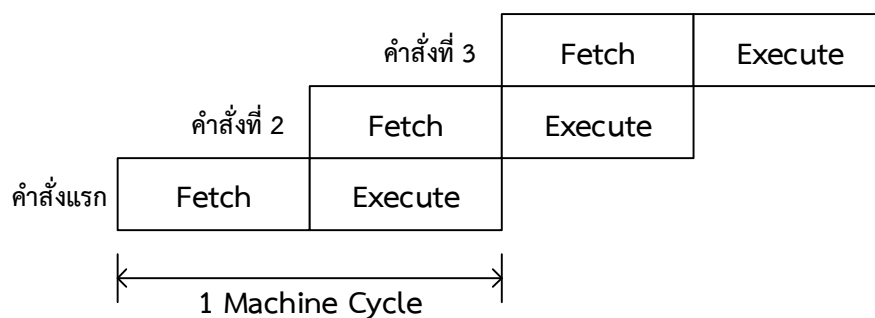
รูปที่ 1.3 ลักษณะการประมวลผลแบบ CISC

จากรูปที่ 1.3 เป็นลักษณะของการประมวลผลแบบ CISC ใน 1 รอบของการประมวลผล ซึ่งจะประกอบไปด้วยขบวนการ Fetch ที่ CPU ทำการอ่านรหัสคำสั่งของโปรแกรมจากหน่วยความจำโปรแกรมแล้วทำการถอดรหัสคำสั่ง หลังจากนั้นจะเข้าสู่ขบวนการ Execute ที่ CPU จะทำการปฏิบัติตามรหัสคำสั่งที่ถอดรหัสได้ให้เสร็จสิ้นเพื่อที่จะได้ทำการ Fetch คำสั่งถัดไปจากหน่วยความจำโปรแกรม

	ใบเนื้อหา	หน้าที่ 4
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

- สถาปัตยกรรมการประมวลผลแบบ RISC (Reduced Instruction Set Computing) เป็นสถาปัตยกรรมการประมวลผลที่ CPU จะประมวลผลชุดคำสั่งด้วยเวลาที่แน่นอนเพียง 1 รอบสัญญาณนาฬิกาต่อคำสั่งเท่านั้น และลดจำนวนคำสั่งให้เหลือเพียงชุดคำสั่งพื้นฐานที่สำคัญ อีกทั้งสร้างรูปแบบกระบวนการถอดรหัสชุดคำสั่งโดยใช้หลักการทำงานส่งผ่านชุดคำสั่งแบบไปป์ไลน์ (Pipeline) คือ การทำงานแบบคาบเกี่ยวกัน (Overlap) โดยการแบ่งหน่วยประมวลผลออกเป็นส่วนย่อย ๆ แล้วแบ่งงานกันรับผิดชอบ โดย CPU ที่ใช้สถาปัตยกรรมแบบ RISC เมื่อเทียบประสิทธิภาพต่อ 1 คำสั่งในการประมวลผลต่อ 1 รอบสัญญาณนาฬิกาจะประมวลผลได้เร็วกว่า CPU ที่ใช้สถาปัตยกรรมแบบ CISC ซึ่งไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมแบบ RISC ได้แก่ตระกูล PIC , ARM และ AVR เป็นต้น



รูปที่ 1.4 ลักษณะการประมวลผลแบบ RISC

จากรูปที่ 1.4 เป็นลักษณะของการประมวลผลแบบ RISC ใน 1 รอบการประมวลผล ซึ่งจากรูปเมื่อ CPU ทำขบวนการ Fetch ของคำสั่งแรกแล้วก็จะทำการเข้าสู่ขบวนการ Execute ของคำสั่งแรกและในขณะเดียวกัน CPU ก็ทำการ Fetch คำสั่งที่ 2 จากหน่วยความจำโปรแกรม และเมื่อ CPU เข้าสู่ขบวนการ Execute ของคำสั่งที่ 2 CPU ก็ทำการ Fetch คำสั่งที่ 3 ซึ่งการประมวลผลของ CPU ที่ใช้สถาปัตยกรรมแบบ RISC ก็จะเป็นลักษณะของขั้นบันไดดังรูปที่ 1.4 ที่เรียกว่าการประมวลผลแบบ Pipeline และเมื่อเทียบกับการประมวลผลแบบ CISC จะสังเกตว่า CPU แบบ RISC จะประมวลผลได้ไวกว่า

1.3 ประเภทของไมโครคอนโทรลเลอร์

ประเภทของไมโครคอนโทรลเลอร์ในหน่วยนี้จะจำแนกประเภทตามจำนวนบิตของข้อมูลที่สามารถประมวลผลได้ในแต่ละครั้งของ CPU ภายในตัวไมโครคอนโทรลเลอร์หรือ Data Bus ซึ่งจะบ่งบอกว่าปกติ 1 พอร์ตของไมโครคอนโทรลเลอร์ตัวนั้นจะมีขาสัญญาณใช้งานได้สูงสุดจำนวนกี่ขาสัญญาณ (อาจจะมองได้ว่า 1 ขาสัญญาณคือข้อมูลจำนวน 1 บิต) โดยปัจจุบันไมโครคอนโทรลเลอร์ที่สามารถนำมาศึกษาและใช้งานได้จะมีด้วยกัน 3 ประเภท ดังลักษณะที่กล่าวข้างต้น คือ

หน้า 5

ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004

หน่วยที่ 1

ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

1. ไมโครคอนโทรลเลอร์ที่มี CPU ในการประมวลผลขนาด 8 บิต และ 1 พอร์ตจะมีขาสัญญาณได้สูงสุดจำนวน 8 ขา เช่นไมโครคอนโทรลเลอร์ตระกูล MCS-51 ,PIC10F ,PIC12F ,PIC16F , ,PIC18F และ AVR เป็นต้น
2. ไมโครคอนโทรลเลอร์ที่มี CPU ในการประมวลผลขนาด 16 บิต และ 1 พอร์ตจะมีขาสัญญาณได้สูงสุดจำนวน 16 ขา เช่นไมโครคอนโทรลเลอร์ตระกูล MSP430 , PIC24F และ dsPIC30F เป็นต้น
3. ไมโครคอนโทรลเลอร์ที่มี CPU ในการประมวลผลขนาด 32 บิต และ 1 พอร์ตจะมีขาสัญญาณได้สูงสุดจำนวน 32 ขา เช่นไมโครคอนโทรลเลอร์ตระกูล ARM7 ,PIC32F ,STM32 ,ESP8285 ,ESP8266 และESP32 เป็นต้น


1.4 คุณสมบัติของไมโครคอนโทรลเลอร์ AT89C51ED2 ,PIC16F887 และ ATMEGA32

เนื่องด้วยในรายวิชานี้ นักศึกษาจะได้ทำการศึกษาการเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 , PIC16F และ AVR โดยไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะเป็นเบอร์ AT89C51ED2 ไมโครคอนโทรลเลอร์ตระกูล PIC16F จะเป็นเบอร์ PIC16F887 และไมโครคอนโทรลเลอร์ตระกูล AVR จะเป็นเบอร์ ATMEGA32 ซึ่งคุณสมบัติพื้นฐานของไมโครคอนโทรลเลอร์เบอร์ต่าง ๆ จะมีข้อมูลดังตารางที่ 1.1

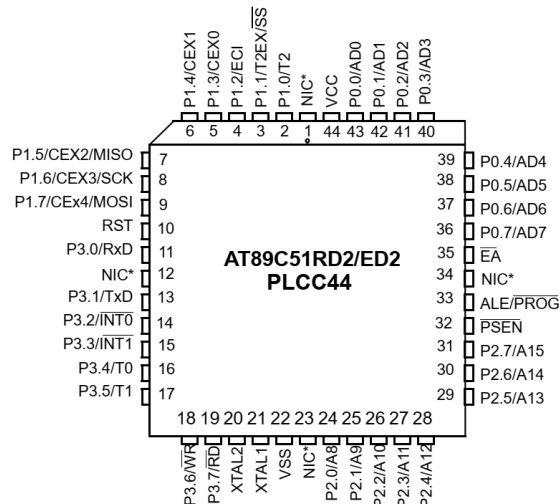
ตารางที่ 1.1 ตารางเปรียบเทียบคุณสมบัติของไมโครคอนโทรลเลอร์ตระกูลต่าง ๆ

ไมโครคอนโทรลเลอร์ เบอร์	ตระกูล	Program Memory	Data Memory	External Memory	EEPROM	I/O	PWM (ch)	SPI (ch)	Timer/ Counter	I2C (ch)	UART (ch)	ADC (ch)
AT89C51ED2	MCS51	64Kbyte	256byte	1792byte	2Kbyte	32	5	1	3	-	1	-
PIC16F887	PIC16F	8Kword	368byte	-	256byte	36	2	1	3	1	1	14
ATMEGA32	AVR	32Kbyte	2Kbyte	-	1Kbyte	32	4	1	3	1	1	8

จากตารางที่ 1.1 ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีจุดเด่นกว่าไมโครคอนโทรลเลอร์ตระกูลอื่น เบอร์อื่น คือมีหน่วยความจำโปรแกรมที่มากถึง 64Kbyte มีขา PWM ที่ใช้สำหรับสร้างสัญญาณพัลส์จำนวน 5 ขา และเป็นไมโครคอนโทรลเลอร์ที่มีเทคโนโลยี ISP (In-System Programming) ที่สามารถดาวน์โหลดโปรแกรมลงตัว อุปกรณ์ไมโครคอนโทรลเลอร์ผ่านพอร์ต UART ร่วมกับโปรแกรม Flip ของบริษัท Atmel โดยไม่ต้องมีเครื่อง โปรแกรมไมโครคอนโทรลเลอร์ ส่วนไมโครคอนโทรลเลอร์ PIC16F887 ของไมโครคอนโทรลเลอร์ตระกูล PIC16F มีจุดเด่นกว่าไมโครคอนโทรลเลอร์ MCS-51 คือสามารถทนต่อสัญญาณรบกวนได้ดีกว่าไมโครคอนโทรลเลอร์ MCS-51 และมีโปรแกรมคอมไพเลอร์ในปัจจุบันที่เป็นแบบ Free ware จากบริษัทผู้ผลิต ประมวลผลได้ไวกว่า รวมถึงมีโมดูล พิเศษในการเชื่อมต่อกับอุปกรณ์ภายนอกได้มากกว่าไมโครคอนโทรลเลอร์ตระกูล MCS-51 ส่วนไมโครคอนโทรลเลอร์ ตระกูล AVR เป็นไมโครคอนโทรลเลอร์ที่ประมวลผลได้ไวที่สุดเมื่อเทียบกับที่สัญญาณนาฬิกาเท่ากันเช่น AT89C51ED2 ต่อ XTAL เท่ากับ 12MHz จะมีความเร็วในการประมวลผลจริงเท่ากับ $12\text{MHz}/12 = 1\text{MHz}$ ถ้าเป็น PIC16F887 ต่อ XTAL เท่ากับ 12MHz จะมีความเร็วในการประมวลผลจริงเท่ากับ $12\text{MHz}/4 = 3\text{MHz}$ ส่วน ATMEGA32 ต่อ XTAL เท่ากับ 12MHz จะมีความเร็วในการประมวลผลจริงเท่ากับ $12\text{MHz}/1 = 12\text{MHz}$ ส่วนความสามารถด้านอื่น ๆ ของไมโครคอนโทรลเลอร์ตระกูล AVR จะใกล้เคียงกับไมโครคอนโทรลเลอร์ PIC แต่ ไมโครคอนโทรลเลอร์ตระกูล AVR จะมีหน่วยความจำโปรแกรมที่ถูกกันไว้ที่เรียกว่า Boot Loader จึงสามารถนำมา สร้างเป็นบอร์ด Arduino ที่ได้รับความนิยมในปัจจุบัน

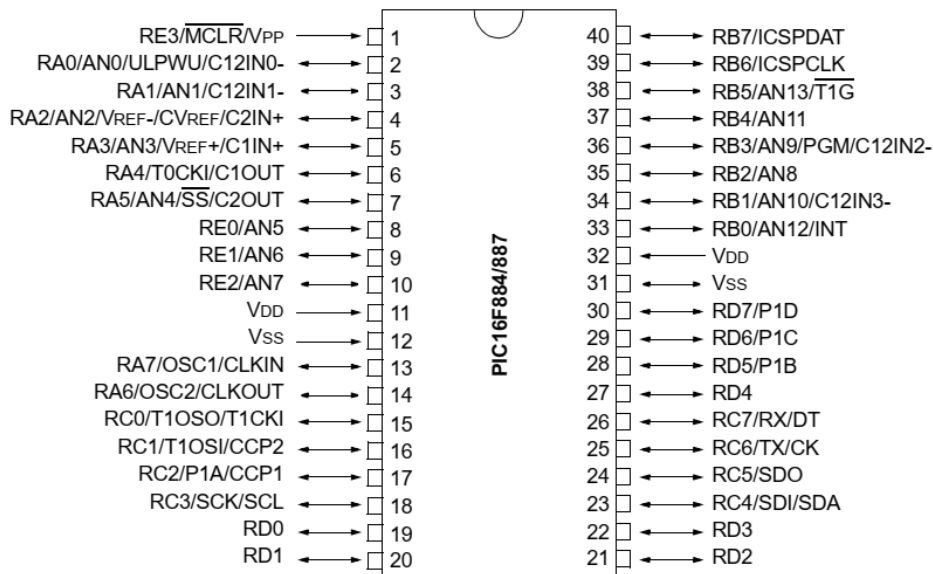
	ใบเนื้อหา	หน้าที่ 6
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.5 รูปตำแหน่งขาสัญญาณและตัวถังของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C51ED2

จากรูปที่ 1.5 เป็นรูปที่แสดงตำแหน่งขาสัญญาณและตัวถังของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C51ED2 ซึ่งมีตัวถังแบบ PLCC 44 ขา มีขาพอร์ตใช้งานเพื่อเชื่อมต่อกับอุปกรณ์อินพุตหรือเอาต์พุต จำนวน 4 พอร์ต ได้แก่พอร์ต P0, P1, P2 และ P3 โดยแต่ละพอร์ตมีขาสัญญาณจำนวน 8 ขา ดังนั้นไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C51ED2 ตัวถังแบบ PLCC 44 ขา มีสัญญาณที่สามารถใช้งานเพื่อเชื่อมต่ออุปกรณ์อินพุตหรือเอาต์พุตทั้งหมด 32 ขา และสามารถ Source Current ทุกขารวมกันได้ไม่เกิน 71mA



รูปที่ 1.6 รูปตำแหน่งขาสัญญาณและตัวถังของไมโครคอนโทรลเลอร์ตระกูล PIC16F เบอร์ PIC16F887

	ใบเนื้อหา		หน้าที่ 7
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

จากรูปที่ 1.6 เป็นรูปที่แสดงตำแหน่งขาสัญญาณและตัวถังของไมโครคอนโทรลเลอร์ตระกูล PIC16F เบอร์ PIC16F887 ซึ่งมีตัวถังแบบ PDIP 40 ขา มีขาพอร์ตใช้งานเพื่อเชื่อมต่อกับอุปกรณ์อินพุตหรือเอาต์พุตจำนวน 5 พอร์ต ได้แก่พอร์ต PORTA ,PORTB , PORTC , PORTD และ PORTE โดยแต่ละพอร์ตจะมีขาสัญญาณดังนี้ PORTA มี 8 ขา ,PORTB มี 8 ขา, PORTC มี 8 ขา, PORTD มี 8 ขา และ PORTE มี 4 ขา ดังนั้นไมโครคอนโทรลเลอร์ตระกูล PIC16F เบอร์ PIC16F887 ตัวถังแบบ PDIP 40 ขา มีสัญญาณที่สามารถใช้งานเพื่อเชื่อมต่ออุปกรณ์อินพุตหรือเอาต์พุตทั้งหมด 36 ขา โดยแต่ละขาสามารถ Sink/Source กระแสได้สูงสุด 25mA

(XCK/T0) PB0	1	40	PA0 (ADC0)
(T1) PB1	2	39	PA1 (ADC1)
(INT2/AIN0) PB2	3	38	PA2 (ADC2)
(OC0/AIN1) PB3	4	37	PA3 (ADC3)
(SS) PB4	5	36	PA4 (ADC4)
(MOSI) PB5	6	35	PA5 (ADC5)
(MISO) PB6	7	34	PA6 (ADC6)
(SCK) PB7	8	33	PA7 (ADC7)
RESET	9	32	AREF
VCC	10	31	GND
GND	11	30	AVCC
XTAL2	12	29	PC7 (TOSC2)
XTAL1	13	28	PC6 (TOSC1)
(RXD) PD0	14	27	PC5 (TDI)
(TXD) PD1	15	26	PC4 (TDO)
(INT0) PD2	16	25	PC3 (TMS)
(INT1) PD3	17	24	PC2 (TCK)
(OC1B) PD4	18	23	PC1 (SDA)
(OC1A) PD5	19	22	PC0 (SCL)
(ICP1) PD6	20	21	PD7 (OC2)

รูปที่ 1.7 รูปตำแหน่งขาสัญญาณและตัวถังของไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ ATMEGA32


จากรูปที่ 1.7 เป็นรูปที่แสดงตำแหน่งขาสัญญาณและตัวถังของไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ ATMEGA32 ซึ่งมีตัวถังแบบ PDIP 40 ขา มีขาพอร์ตใช้งานเพื่อเชื่อมต่อกับอุปกรณ์อินพุตหรือเอาต์พุตจำนวน 4 พอร์ต ได้แก่พอร์ต PA ,PB ,PC และ PD โดยแต่ละพอร์ตมีขาสัญญาณจำนวน 8 ขา ดังนั้นไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ ATMEGA32 ตัวถังแบบ PDIP 40 ขา มีสัญญาณที่สามารถใช้งานเพื่อเชื่อมต่ออุปกรณ์อินพุตหรือเอาต์พุตทั้งหมด 32 ขา โดยแต่ละขาสามารถ Sink/Source กระแสได้สูงสุดประมาณ 20mA

1.5 การใช้งานขาสัญญาณของไมโครคอนโทรลเลอร์


การใช้งานขาของไมโครคอนโทรลเลอร์เพื่อติดต่อกับอุปกรณ์อินพุตและเอาต์พุตจะมี 2 ลักษณะด้วยกัน คือ

1. ขาสัญญาณของไมโครคอนโทรลเลอร์ที่เมื่อมีการใช้งานจะปรับสภาพขาสัญญาณเองอัตโนมัติเพื่อทำหน้าที่เป็นขาสัญญาณดิจิตอลแบบอินพุตหรือเอาต์พุต ซึ่งขึ้นอยู่กับชุดคำสั่งที่ติดต่อกับขาสัญญาณของไมโครคอนโทรลเลอร์ โดยเรียกขาสัญญาณของไมโครคอนโทรลเลอร์แบบนี้ว่าขาสัญญาณแบบ Bi-directional และไมโครคอนโทรลเลอร์แบบนี้จะมีรีจิสเตอร์ที่ใช้งานเพื่ออ่านค่าหรือเขียนค่าข้อมูลกับขาสัญญาณของพอร์ตไมโครคอนโทรลเลอร์เพียง 1 ตัวเท่านั้น เช่นขาพอร์ตของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นต้น โดยมีตัวอย่างการใช้งานชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์เพื่อใช้งานขาสัญญาณดังนี้

	ใบเนื้อหา		หน้าที่ 8	
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1	
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์			
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์				
<p>MOV A,P1 ; หมายถึงให้อ่านข้อมูลจากขาสัญญาณของพอร์ต P1 ไปเก็บไว้ที่รีจิสเตอร์ A</p> <p>MOV P1,A ; หมายถึงให้นำค่าข้อมูลในรีจิสเตอร์ A ส่งออกไปที่ขาสัญญาณของพอร์ต P1</p> <p>ขาสัญญาณของพอร์ตไมโครคอนโทรลเลอร์ MCS-51 ทุกขาเสมือนมีการต่อตัวต้านทาน Pull-up ภายใน ยกเว้นพอร์ต P0 เมื่อมีการใช้งานจะต้องทำการต่อตัวต้านทาน Pull-up ภายนอก</p> <p>2. ขาสัญญาณของไมโครคอนโทรลเลอร์ที่เมื่อต้องการให้ทำหน้าที่เป็นขาสัญญาณดิจิทัลแบบอินพุตหรือเอาต์พุต จะต้องทำการกำหนดทิศทางของขาสัญญาณก่อน จึงจะสามารถใช้งานขาสัญญาณของพอร์ตนั้น ๆ เป็นขาสัญญาณอินพุตหรือเอาต์พุต ซึ่ง 1 พอร์ตจะมีรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานอย่างน้อย 2 ตัว โดยตัวแรกจะเป็นรีจิสเตอร์ที่กำหนดทิศทางของขาสัญญาณว่าจะทำหน้าที่เป็นอินพุตหรือเอาต์พุต ส่วนรีจิสเตอร์ที่เหลือเป็นรีจิสเตอร์ที่ใช้สำหรับติดต่อกับอุปกรณ์อินพุตหรือเอาต์พุต เช่นขาสัญญาณของไมโครคอนโทรลเลอร์ตระกูล PIC16F และ AVR เป็นต้น โดยจะมีตารางเปรียบเทียบรีจิสเตอร์ของไมโครคอนโทรลเลอร์ตระกูล PIC16F และ AVR ดังตารางที่ 1.2</p> <p>ตารางที่ 1.2 ตารางเปรียบเทียบรีจิสเตอร์ของไมโครคอนโทรลเลอร์ตระกูล MCS51 , PIC16F และ AVR ที่เกี่ยวกับการใช้งานขาสัญญาณพอร์ตเพื่อติดต่อกับอุปกรณ์อินพุตหรือเอาต์พุต</p>				
ไมโครคอนโทรลเลอร์ตระกูล	เบอร์ของอุปกรณ์	รีจิสเตอร์ที่ทำหน้าที่กำหนดทิศทางขาสัญญาณ	รีจิสเตอร์ที่ทำหน้าที่ติดต่อกับอุปกรณ์อินพุต	รีจิสเตอร์ที่ทำหน้าที่ติดต่อกับอุปกรณ์เอาต์พุต
MCS-51	AT89C51ED2	-	Px	Px
PIC16F	PIC16F887	TRISx	PORTx	PORTx
AVR	ATMEGA32	DDRx	PINx	PORTx
<p>หมายเหตุ สำหรับไมโครคอนโทรลเลอร์เบอร์ AT89C51ED2 ค่า x คือตำแหน่งของพอร์ตได้แก่ 0,1,2 และ3</p> <p>สำหรับไมโครคอนโทรลเลอร์เบอร์ PIC16F887 ค่า x คือตำแหน่งของพอร์ตได้แก่ A,B,C,D และE</p> <p>สำหรับไมโครคอนโทรลเลอร์เบอร์ ATMEGA32 ค่า x คือตำแหน่งของพอร์ตได้แก่ A,B,C และD</p> <p>ตัวอย่างการใช้งานชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ PIC16F887 เพื่อกำหนดทิศทางของขาสัญญาณพอร์ตเพื่อติดต่อกับอุปกรณ์อินพุตและเอาต์พุต</p> <p>MOVLW 0x0F ; ให้รีจิสเตอร์ W เก็บค่าข้อมูลคงที่ค่า 0x0F</p> <p>MOVWF TRISD ; ให้รีจิสเตอร์ TRISD เก็บค่าข้อมูลจากรีจิสเตอร์ W คือค่า 0x0F หรือค่า 0b00001111 มีความหมายว่าให้ขาสัญญาณ PORTD ขา RD4 – RD7 ให้ทำหน้าที่เป็นขาสัญญาณเอาต์พุต ส่วนขา RD0 – RD3 ให้ทำหน้าที่เป็นขาสัญญาณอินพุต</p> <p>MOVF PORTD,W ; ให้รีจิสเตอร์ W เก็บค่าข้อมูลจากขาสัญญาณ RD0 – RD3</p> <p>MOVWF PORTD ; ให้ขา PORTD ตำแหน่งที่ขา RD4 – RD7 มีค่าข้อมูลเท่ากับรีจิสเตอร์ W ที่ตำแหน่งบิต D4 – D7</p>				

	ใบเนื้อหา	หน้าที่ 9
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

<p>ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์</p> <p>ตัวอย่างการใช้งานชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ ATMEGA32 เพื่อกำหนดทิศทางของขาสัญญาณพอร์ตเพื่อติดต่อกับอุปกรณ์อินพุตและเอาต์พุต</p> <p>LDI R16,0x0F ; ให้รีจิสเตอร์ R16 เก็บค่าข้อมูลคงที่ค่า 0x0F</p> <p>OUT DDRD,R16 ; ให้รีจิสเตอร์ DDRD เก็บค่าข้อมูลจากรีจิสเตอร์ R16 คือค่า 0x0F หรือค่า 0b00001111 มีความหมายว่าให้ขาสัญญาณ PD ขา PD4 – PD7 ให้ทำหน้าที่เป็นขาสัญญาณอินพุต ส่วนขา PD0 – PD3 ให้ทำหน้าที่เป็นขาสัญญาณเอาต์พุต</p> <p>IN R16,PIND ; ให้รีจิสเตอร์ R16 เก็บค่าข้อมูลจากขาสัญญาณ PD4 – PD7</p> <p>OUT PORTD,R16 ; ให้ขา PORTD ตำแหน่งที่ขา PD0 – PD3 มีค่าข้อมูลเท่ากับรีจิสเตอร์ R16 ที่ตำแหน่งบิต D0 – D3</p> <p>2. ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์</p> <p>ภาษาแอสเซมบลี (Assembly Language) หมายถึง ภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่งซึ่งจะทำงานโดยขึ้นกับรุ่นของไมโครโพรเซสเซอร์ หรือ "หน่วยประมวลผล" (CPU) ของไมโครคอนโทรลเลอร์บอร์ดนั้น ๆ การใช้ภาษาแอสเซมบลีจำเป็นต้องผ่านการแปลภาษาด้วยคอมไพเลอร์เฉพาะเรียกว่า แอสเซมเบลอร์ (assembler) ให้อยู่ในรูปของรหัสคำสั่งก่อน (เช่น .hex) โดยปกติภาษานี้ค่อนข้างมีความยุ่งยากในการใช้งาน และการเขียนโปรแกรมเป็นจำนวนบรรทัดที่มากเมื่อเปรียบเทียบกับการใช้ภาษาระดับสูง เช่น ภาษา C หรือภาษา BASIC แต่จะทำให้ได้ผลลัพธ์การทำงานของโปรแกรมเร็วกว่า และขนาดของตัวโปรแกรมที่ใช้งานจริงที่เป็นภาษาเครื่อง (opcode หรือ hex file) มีขนาดเนื้อที่น้อยกว่าโปรแกรมที่สร้างจากภาษาอื่นมาก จึงนิยมใช้ภาษานี้เมื่อต้องการประหยัดเวลาทำงานของไมโครคอนโทรลเลอร์ และเพิ่มประสิทธิภาพของโปรแกรม เนื่องจากตัวคำสั่งภายในภาษาอังกฤษเฉพาะกับรุ่นของหน่วยประมวลผล ดังนั้นถ้ามีการเปลี่ยนแปลงไปใช้กับหน่วยประมวลผลอื่นหรือระบบอื่นจะต้องมีการปรับแก้ตัวคำสั่งภายใน ซึ่งบางครั้งอาจไม่สามารถปรับปรุงแก้ไขได้อย่างสมบูรณ์</p> <p>2.1 ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล MCS-51</p> <p>ไมโครคอนโทรลเลอร์ MCS-51 มีรูปแบบชุดคำสั่งภาษาแอสเซมบลีทั้งหมดประมาณ 111 รูปแบบชุดคำสั่ง และแต่ละรูปแบบชุดคำสั่งจะมีขนาดของข้อมูลที่ไม่เท่ากัน โดยสามารถแบ่งออกเป็นกลุ่มได้ 5 กลุ่ม ดังนี้</p> <ol style="list-style-type: none">1. กลุ่มคำสั่งการโอนย้ายข้อมูล (Data Transfer Instructions) มี 28 รูปแบบชุดคำสั่ง2. กลุ่มคำสั่งทำงานคณิตศาสตร์ (Arithmetic Operation instructions) มี 24 รูปแบบชุดคำสั่ง3. กลุ่มคำสั่งทำงานลอจิก (Logical Operation Instructions) มี 25 รูปแบบชุดคำสั่ง4. กลุ่มคำสั่งจัดการข้อมูลระดับบิต (Boolean Variable Manipulated Instructions) มี 17 รูปแบบชุดคำสั่ง5. กลุ่มคำสั่งการกระโดด (Program Branching Instructions) มี 17 รูปแบบชุดคำสั่ง


	ใบเนื้อหา		หน้าที่ 10
	ชื่อวิชา	ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

R_n	Register R7-R0 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
@ R_i	8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.
#data	8-bit constant included in instruction.
#data 16	16-bit constant included in instruction.
addr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K byte Program Memory address space.
addr 11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
bit	Direct Addressed bit in Internal Data RAM or Special Function Register.

รูปที่ 1.8 เป็นรูปแสดงตารางอธิบายความหมายของข้อความหรือสัญลักษณ์ที่ใช้ในชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะมีรีจิสเตอร์หลักที่ทำหน้าที่ในการเก็บค่าตัวตั้งของการประมวลผลทางคณิตศาสตร์และลอจิก คือรีจิสเตอร์ A และยังเป็นรีจิสเตอร์ที่ทำหน้าที่ในการเก็บผลลัพธ์ด้วย โดยรีจิสเตอร์ A จะเป็นรีจิสเตอร์ขนาด 8 บิต ส่วนข้อมูลที่นำมากระทำกับรีจิสเตอร์ A อาจจะเป็นค่าข้อมูลคงที่ขนาด 8 บิต ซึ่งจะต้องเขียนในรูปแบบ #data หรือข้อมูลที่นำมากระทำกับรีจิสเตอร์ A อาจจะเป็นค่าข้อมูลที่เก็บอยู่ในรีจิสเตอร์แบบที่ R_n หรือข้อมูลที่นำมากระทำกับรีจิสเตอร์ A อาจจะเป็นค่าข้อมูลที่เก็บอยู่ในตำแหน่งหน่วยความจำข้อมูลที่สามารถเข้าถึงได้แบบโดยตรงที่ตำแหน่ง direct เป็นต้น ส่วนการประมวลผลทางด้านคณิตศาสตร์ในรูปแบบของการคูณ และหาร จะใช้รีจิสเตอร์ A เป็นตัวเก็บค่าข้อมูลตัวตั้งกับผลลัพธ์ และใช้รีจิสเตอร์ B ในการเก็บค่าข้อมูลตัวกระทำกับผลลัพธ์ ส่วนรีจิสเตอร์ขนาด 16 บิต ที่ผู้ใช้งานสามารถนำไปใช้ในการประมวลผลหรือโอนย้ายข้อมูลจะมีเพียง 1 ตัวเท่านั้น สำหรับไมโครคอนโทรลเลอร์ MCS-51 คือรีจิสเตอร์ DPTR

	ใบเนื้อหา		หน้าที่ 11
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Mnemonic	Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS			
ADD A,R _n	Add register to Accumulator	1	12
ADD A,direct	Add direct byte to Accumulator	2	12
ADD A,@R _i	Add indirect RAM to Accumulator	1	12
ADD A,#data	Add immediate data to Accumulator	2	12
ADDC A,R _n	Add register to Accumulator with Carry	1	12
ADDC A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC A,@R _i	Add indirect RAM to Accumulator with Carry	1	12
ADDC A,#data	Add immediate data to Acc with Carry	2	12
SUBB A,R _n	Subtract Register from Acc with borrow	1	12
SUBB A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB A,@R _i	Subtract indirect RAM from ACC with borrow	1	12
SUBB A,#data	Subtract immediate data from Acc with borrow	2	12
INC A	Increment Accumulator	1	12
INC R _n	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @R _i	Increment direct RAM	1	12
DEC A	Decrement Accumulator	1	12
DEC R _n	Decrement Register	1	12
DEC direct	Decrement direct byte	2	12
DEC @R _i	Decrement indirect RAM	1	12
INC DPTR	Increment Data Pointer	1	24
MUL AB	Multiply A & B	1	48
DIV AB	Divide A by B	1	48
DA A	Decimal Adjust Accumulator	1	12

Note: 1. All mnemonics copyrighted © Intel Corp., 1980.

Mnemonic	Description	Byte	Oscillator Period
LOGICAL OPERATIONS			
ANL A,R _n	AND Register to Accumulator	1	12
ANL A,direct	AND direct byte to Accumulator	2	12
ANL A,@R _i	AND indirect RAM to Accumulator	1	12
ANL A,#data	AND immediate data to Accumulator	2	12
ANL direct,A	AND Accumulator to direct byte	2	12
ANL direct,#data	AND immediate data to direct byte	3	24
ORL A,R _n	OR register to Accumulator	1	12
ORL A,direct	OR direct byte to Accumulator	2	12
ORL A,@R _i	OR indirect RAM to Accumulator	1	12
ORL A,#data	OR immediate data to Accumulator	2	12
ORL direct,A	OR Accumulator to direct byte	2	12
ORL direct,#data	OR immediate data to direct byte	3	24
XRL A,R _n	Exclusive-OR register to Accumulator	1	12
XRL A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL A,@R _i	Exclusive-OR indirect RAM to Accumulator	1	12
XRL A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR A	Clear Accumulator	1	12
CPL A	Complement Accumulator	1	12
RL A	Rotate Accumulator Left	1	12
RLC A	Rotate Accumulator Left through the Carry	1	12
LOGICAL OPERATIONS (continued)			

รูปที่ 1.9 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ MCS-51

ใบเนื้อหา

หน้าที่ 12

ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004

หน่วยที่ 1

ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์


ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Mnemonic		Description	Byte	Oscillator Period
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,R _n	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@R _i	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	R _n ,A	Move Accumulator to register	1	12
MOV	R _n ,direct	Move direct byte to register	2	24
MOV	R _n ,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,R _n	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@R _i	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@R _i ,A	Move Accumulator to indirect RAM	1	12
MOV	@R _i ,direct	Move direct byte to indirect RAM	2	24
MOV	@R _i ,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@R _i	Move External RAM (8-bit addr) to Acc	1	24
DATA TRANSFER (continued)				
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24

Mnemonic		Description	Byte	Oscillator Period
MOVX	@R _i ,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R _n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R _i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R _i	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24

รูปที่ 1.20 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ MCS-51 (ต่อ)

รูปที่ 1.20 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ MCS-51 (ต่อ)

	ใบเนื้อหา		หน้าที่ 13
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์


Mnemonic		Description	Byte	Oscillator Period
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	R _n ,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@R _i ,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	R _n ,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

รูปที่ 1.21 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ MCS-51 (ต่อ)

2.2 ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล PIC16F

ไมโครคอนโทรลเลอร์ PIC16F มีรูปแบบชุดคำสั่งภาษาแอสเซมบลีทั้งหมดประมาณ 35 รูปแบบ ชุดคำสั่งและแต่ละรูปแบบชุดคำสั่งจะมีขนาดของข้อมูลเท่ากันที่ 14 บิต โดยสามารถแบ่งออกเป็นกลุ่มได้ 3 กลุ่ม ดังนี้

1. กลุ่มคำสั่งกระทำกับข้อมูลแบบไบต์ (Byte-Oriented File Register Operation) มี 18 รูปแบบชุดคำสั่ง
2. กลุ่มคำสั่งกระทำกับข้อมูลแบบบิต (Bit-Oriented File Register Operation) มี 4 รูปแบบชุดคำสั่ง
3. กลุ่มคำสั่งกระทำกับค่าข้อมูลคงที่ (Literal and Control Operation) มี 13 รูปแบบชุดคำสั่ง


	ใบเนื้อหา		หน้าที่ 14
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in w, d = 1: store result in file register f. Default is d = 1 .
PC	Program Counter
\overline{TO}	Time-out bit
C	Carry bit
DC	Digit carry bit
Z	Zero bit
\overline{PD}	Power-down bit

รูปที่ 1.22 เป็นรูปแสดงตารางอธิบายความหมายของข้อความหรือสัญลักษณ์ที่ใช้ในชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ PIC16F

ไมโครคอนโทรลเลอร์ตระกูล PIC16F จะมีรีจิสเตอร์หลักที่ทำหน้าที่ในการเก็บค่าตัวตั้งของการประมวลผลทางคณิตศาสตร์และลอจิก คือรีจิสเตอร์ W ส่วนผลลัพธ์ของการประมวลผลสามารถเลือกได้ว่าจะเก็บไว้ที่รีจิสเตอร์ W หรือตำแหน่งของรีจิสเตอร์ตัวกระทำตามตารางในรูปที่ 1.22


	ใบเนื้อหา		หน้าที่ 15
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Mnemonic, Operands	Description	Cycles	14-Bit Opcode				Status Affected	Notes	
			MSb		LSb				
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	1, 2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1, 2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRWF	—	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1, 2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1, 2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2, 3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2, 3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1, 2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1, 2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	—	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1, 2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1, 2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	1, 2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1, 2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1, 2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1, 2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C, DC, Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	00	0000	0110	0100	\overline{TO} , \overline{PD}	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	—	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	—	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	—	Go into Standby mode	1	00	0000	0110	0011	\overline{TO} , \overline{PD}	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C, DC, Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	


- Note 1:** When an I/O register is modified as a function of itself (e.g., MOVF GPIO, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.
- 3:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

รูปที่ 1.23 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ PIC16F

	ใบเนื้อหา		หน้าที่ 16
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์																																									
<div>2.3 ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล AVR</div> <div>ไมโครคอนโทรลเลอร์ AVR มีรูปแบบชุดคำสั่งภาษาแอสเซมบลีทั้งหมดประมาณ 131 รูปแบบชุดคำสั่ง โดยสามารถแบ่งออกเป็นกลุ่มได้ 5 กลุ่ม ดังนี้</div> <div><div>1. กลุ่มคำสั่งกระทำทางคณิตศาสตร์และลอจิก (Arithmetic and Logic Instructions) มี 28 รูปแบบชุดคำสั่ง</div><div>2. กลุ่มคำสั่งการกระโดด (Branch Instructions) มี 36 รูปแบบชุดคำสั่ง</div><div>3. กลุ่มคำสั่งการโอนย้ายข้อมูล (Data transfer Instructions) มี 35 รูปแบบชุดคำสั่ง</div><div>4. กลุ่มคำสั่งการกระโดด (Bit and Bit-Test Instructions) มี 28 รูปแบบชุดคำสั่ง</div><div>5. กลุ่มคำสั่งการโอนย้ายข้อมูล (MCU control Instructions) มี 4 รูปแบบชุดคำสั่ง</div></div> <div><div>Status Register (SREG)</div><table><tr><td>SREG</td><td>Status Register</td></tr><tr><td>C</td><td>Carry Flag</td></tr><tr><td>Z</td><td>Zero Flag</td></tr><tr><td>N</td><td>Negative Flag</td></tr><tr><td>V</td><td>Two's complement overflow indicator</td></tr><tr><td>S</td><td>$N \oplus V$, for signed tests</td></tr><tr><td>H</td><td>Half Carry Flag</td></tr><tr><td>T</td><td>Transfer bit used by BLD and BST instructions</td></tr><tr><td>I</td><td>Global Interrupt Enable/Disable Flag</td></tr></table><div>Registers and Operands</div><table><tr><td>Rd:</td><td>Destination (and source) register in the Register File</td></tr><tr><td>Rr:</td><td>Source register in the Register File</td></tr><tr><td>R:</td><td>Result after instruction is executed</td></tr><tr><td>K:</td><td>Constant data</td></tr><tr><td>k:</td><td>Constant address</td></tr><tr><td>b:</td><td>Bit in the Register File or I/O Register (3-bit)</td></tr><tr><td>s:</td><td>Bit in the Status Register (3-bit)</td></tr><tr><td>X,Y,Z:</td><td>Indirect Address Register (X=R27:R26, Y=R29:R28, and Z=R31:R30)</td></tr><tr><td>A:</td><td>I/O location address</td></tr><tr><td>q:</td><td>Displacement for direct addressing (6-bit)</td></tr></table></div>				SREG	Status Register	C	Carry Flag	Z	Zero Flag	N	Negative Flag	V	Two's complement overflow indicator	S	$N \oplus V$, for signed tests	H	Half Carry Flag	T	Transfer bit used by BLD and BST instructions	I	Global Interrupt Enable/Disable Flag	Rd:	Destination (and source) register in the Register File	Rr:	Source register in the Register File	R:	Result after instruction is executed	K:	Constant data	k:	Constant address	b:	Bit in the Register File or I/O Register (3-bit)	s:	Bit in the Status Register (3-bit)	X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28, and Z=R31:R30)	A:	I/O location address	q:	Displacement for direct addressing (6-bit)
SREG	Status Register																																								
C	Carry Flag																																								
Z	Zero Flag																																								
N	Negative Flag																																								
V	Two's complement overflow indicator																																								
S	$N \oplus V$, for signed tests																																								
H	Half Carry Flag																																								
T	Transfer bit used by BLD and BST instructions																																								
I	Global Interrupt Enable/Disable Flag																																								
Rd:	Destination (and source) register in the Register File																																								
Rr:	Source register in the Register File																																								
R:	Result after instruction is executed																																								
K:	Constant data																																								
k:	Constant address																																								
b:	Bit in the Register File or I/O Register (3-bit)																																								
s:	Bit in the Status Register (3-bit)																																								
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28, and Z=R31:R30)																																								
A:	I/O location address																																								
q:	Displacement for direct addressing (6-bit)																																								

รูปที่ 1.24 เป็นรูปอธิบายความหมายของข้อความหรือสัญลักษณ์ที่ใช้ในชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR

	ใบเนื้อหา		หน้าที่ 17
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	Rd,K	Add Immediate to Word	$RdH:RdL \leftarrow RdH:RdL + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIV	Rd,K	Subtract Immediate from Word	$RdH:RdL \leftarrow RdH:RdL - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \cdot Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \cdot K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \text{SFF} - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \cdot (\text{SFF} - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \cdot Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow \text{SFF}$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow \text{Stack}$	None	4
RETI		Interrupt Return	$PC \leftarrow \text{Stack}$	I	4
CPSE	Rd,Rr	Compare, Skip If Equal	If $(Rd = Rr)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
CP	Rd,Rr	Compare	$Rd - Rr$	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	$Rd - K$	Z, N,V,C,H	1
SBRC	Rr, b	Skip If Bit in Register Cleared	If $(Rr(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBRSC	Rr, b	Skip If Bit in Register Is Set	If $(Rr(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIC	P, b	Skip If Bit in I/O Register Cleared	If $(P(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBSIS	P, b	Skip If Bit in I/O Register Is Set	If $(P(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1/2/3
BRBS	s, k	Branch If Status Flag Set	If $(SREG(s) = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRBC	s, k	Branch If Status Flag Cleared	If $(SREG(s) = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BREQ	k	Branch If Equal	If $(Z = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRNE	k	Branch If Not Equal	If $(Z = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRCS	k	Branch If Carry Set	If $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRCC	k	Branch If Carry Cleared	If $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRSH	k	Branch If Same or Higher	If $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRLO	k	Branch If Lower	If $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRMI	k	Branch If Minus	If $(N = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRPL	k	Branch If Plus	If $(N = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRGE	k	Branch If Greater or Equal, Signed	If $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRLT	k	Branch If Less Than Zero, Signed	If $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHS	k	Branch If Half Carry Flag Set	If $(H = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRHC	k	Branch If Half Carry Flag Cleared	If $(H = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTS	k	Branch If T Flag Set	If $(T = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRTC	k	Branch If T Flag Cleared	If $(T = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRVS	k	Branch If Overflow Flag Is Set	If $(V = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2
BRVC	k	Branch If Overflow Flag Is Cleared	If $(V = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2

รูปที่ 1.25 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR

ใบเนื้อหา

หน้าที่ 18

ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004

หน่วยที่ 1


ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch If Interrupt Enabled	If (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch If Interrupt Disabled	If (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, -X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, -Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z + 1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q, Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z + 1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	Stack ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← Stack	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P, b	Set Bit in I/O Register	IO(P, b) ← 1	None	2
CBI	P, b	Clear Bit in I/O Register	IO(P, b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z, C, N, V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z, C, N, V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z, C, N, V	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z, C, N, V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z, C, N, V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ← Rd(7..4), Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1

รูปที่ 1.26 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR (ต่อ)

รูปที่ 1.26 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR (ต่อ)

	ใบเนื้อหา		หน้าที่ 19
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear Half Carry Flag in SREG	$H \leftarrow 0$	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A

รูปที่ 1.27 เป็นรูปแสดงตารางชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR (ต่อ)

ไมโครคอนโทรลเลอร์ตระกูล AVR จะมีรีจิสเตอร์หลักที่ทำหน้าที่ในการเก็บค่าตัวตั้งของการประมวลผลทางคณิตศาสตร์และลอจิก คือรีจิสเตอร์ Rd (R0 – R31) และยังเป็นรีจิสเตอร์ที่ทำหน้าที่ในการเก็บผลลัพธ์ด้วย โดยรีจิสเตอร์ Rd จะเป็นรีจิสเตอร์ขนาด 8 บิต ส่วนข้อมูลที่นำมากระทำกับรีจิสเตอร์ Rd อาจจะเป็นค่าข้อมูลคงที่ขนาด 8 บิต ซึ่งจะต้องเขียนในรูปแบบ K หรือข้อมูลที่นำมากระทำกับรีจิสเตอร์ Rd อาจจะเป็นค่าข้อมูลที่เก็บอยู่ในรีจิสเตอร์ใช้งานทั่วไปที่ Rr (R0 – R31) เป็นต้น ส่วนรีจิสเตอร์ขนาด 16 บิต ที่ผู้ใช้งานสามารถนำไปใช้ในการประมวลผลหรือโอนย้ายข้อมูลจะมี 3 ตัวได้แก่ X (R27:R26) , Y (R29:R28) และ Z (R31:R30)

2.4 ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ที่ใช้สำหรับการเขียนโปรแกรมเพื่อจำลองการทำงานเป็นลอจิกเกต

ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ MCS-51 ,PIC และ AVR ที่ใช้สำหรับการเขียนโปรแกรมเพื่อจำลองการทำงานเป็นลอจิกเกต ส่วนใหญ่จะเป็นกลุ่มคำสั่งที่ใช้ในการจัดการกับข้อมูลระดับบิตดังต่อไปนี้

2.4.1 กลุ่มคำสั่งจัดการข้อมูลระดับบิตของไมโครคอนโทรลเลอร์ MCS-51 ได้แก่

CLR C ; คือการทำให้บิต Carry มีค่าเป็นลอจิก '0'

CLR bit ; คือการทำให้ตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตมีค่าเป็นลอจิก '0'


SETB C ; คือการทำให้บิต Carry มีค่าเป็นลอจิก '1'

SETB bit ; คือการทำให้ตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตมีค่าเป็นลอจิก '1'


CPL C ; คือการทำให้บิต Carry มีค่าเป็นลอจิกตรงกันข้ามกับค่าปัจจุบัน


CPL bit ; คือการทำให้ตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตมีค่าเป็นลอจิกตรงกันข้ามกับค่าปัจจุบัน

ANL C,bit ; คือนำค่าข้อมูลของบิต Carry มาทำการ AND กับค่าข้อมูลในตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตแล้วนำผลลัพธ์ที่ได้เก็บไว้ที่บิต Carry

	ใบเนื้อหา	หน้าที่ 20
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์
<p>ANL C,/bit ; คือนำค่าข้อมูลของบิต Carry มาทำการ AND กับค่าข้อมูลตรงกันข้ามกับข้อมูลปัจจุบันในตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตแล้วนำผลลัพธ์ที่ได้เก็บไว้ที่บิต Carry</p> <p>ORL C,/bit ; คือนำค่าข้อมูลของบิต Carry มาทำการ OR กับค่าข้อมูลในตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตแล้วนำผลลัพธ์ที่ได้เก็บไว้ที่บิต Carry</p> <p>ORL C,/bit ; คือนำค่าข้อมูลของบิต Carry มาทำการ OR กับค่าข้อมูลตรงกันข้ามกับข้อมูลปัจจุบันในตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตแล้วนำผลลัพธ์ที่ได้เก็บไว้ที่บิต Carry</p> <p>MOV C,/bit ; คือการนำค่าข้อมูลในตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตไปเก็บไว้ที่บิต Carry</p> <p>MOV bit,C ; คือนำค่าข้อมูลของบิต Carry ไปเก็บไว้ในตำแหน่งบิตของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิต</p> <p>JC rel ; กระโดดไปยังแอดเดรสในระยยะสัมพันธ์เมื่อบิต Carry มีค่าข้อมูลเป็นลอจิก ‘1’</p> <p>JNC rel ; กระโดดไปยังแอดเดรสในระยยะสัมพันธ์เมื่อบิต Carry มีค่าข้อมูลเป็นลอจิก ‘0’</p> <p>JB bit,rel ; กระโดดไปยังแอดเดรสในระยยะสัมพันธ์เมื่อบิตในตำแหน่งของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตมีค่าข้อมูลเป็นลอจิก ‘1’</p> <p>JNB bit,rel ; กระโดดไปยังแอดเดรสในระยยะสัมพันธ์เมื่อบิตในตำแหน่งของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตมีค่าข้อมูลเป็นลอจิก ‘0’</p> <p>JBC bit,rel ; กระโดดไปยังแอดเดรสในระยยะสัมพันธ์เมื่อบิตในตำแหน่งของหน่วยความจำข้อมูลที่สามารถเข้าถึงได้ในระดับบิตมีค่าข้อมูลเป็นลอจิก ‘1’ หลังจากนั้นก็ทำให้ค่าข้อมูลของบิตนั้นมีค่าเป็นลอจิก ‘0’</p> <p>2.4.2 กลุ่มคำสั่งจัดการข้อมูลระดับบิตของไมโครคอนโทรลเลอร์ PIC16F ได้แก่</p> <p>BCF f,b ; ทำให้บิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำข้อมูล f มีค่าเป็นลอจิก ‘0’</p> <p>BSF f,b ; ทำให้บิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำข้อมูล f มีค่าเป็นลอจิก ‘1’</p> <p>BTFSC f,b ; ทำการตรวจสอบบิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำข้อมูล f ถ้ามีค่าเป็นลอจิก ‘0’ ให้ข้ามการประมวลผลไป 1 คำสั่ง</p> <p>BTFSS f,b ; ทำการตรวจสอบบิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำข้อมูล f ถ้ามีค่าเป็นลอจิก ‘1’ ให้ข้ามการประมวลผลไป 1 คำสั่ง</p> <p>2.4.3 กลุ่มคำสั่งจัดการข้อมูลระดับบิตของไมโครคอนโทรลเลอร์ AVR ได้แก่</p> <p>CBI P,b ; ทำให้บิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำ I/O มีค่าเป็นลอจิก ‘0’</p> <p>SBI P,b ; ทำให้บิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำ I/O มีค่าเป็นลอจิก ‘1’</p> <p>SBIC P,b ; ทำการตรวจสอบบิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำ I/O ถ้ามีค่าเป็นลอจิก ‘0’ ให้ข้ามการประมวลผลไป 1 คำสั่ง</p>

	ใบเนื้อหา		หน้าที่ 21																	
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1																	
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์																			
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์																				
SBIS P,b ; ทำการตรวจสอบบิตของข้อมูล ณ ตำแหน่งที่ b ของหน่วยความจำ I/O ถ้ามีค่าเป็นลอจิก '1' ให้ห้ามการประมวลผลไป 1 คำสั่ง																				
2.5 การเขียนโปรแกรมภาษาแอสเซมบลี																				
<p>การเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ในแต่ละตระกูลจะมีหลักการเดียวกัน คือ จะต้องเขียนโปรแกรมด้วยภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูลนั้น ๆ โดยใช้โปรแกรม Text Editor ตัวใดก็ได้แล้วทำการบันทึกลงเป็นนามสกุล .asm หลังจากนั้นก็ทำการแปลงไฟล์โปรแกรมภาษาแอสเซมบลีให้เป็นไฟล์ที่ประกอบไปด้วยข้อมูลแอดเดรสเริ่มต้นของโปรแกรมและรหัสคำสั่งของภาษาแอสเซมบลีที่อยู่ในรูปของเลขฐานสิบหกเรียงต่อกัน ซึ่งจะเรียกว่าไฟล์นามสกุล .hex เพื่อนำเอาไฟล์ข้อมูลนี้ไปโปรแกรมลงบนชิปไมโครคอนโทรลเลอร์ โดยขั้นตอนการแปลงไฟล์นามสกุล .asm เป็น .hex จะเรียกว่าการแอสเซมเบลด้วยโปรแกรม assembler ของไมโครคอนโทรลเลอร์ตระกูลนั้น ๆ ซึ่งโดยส่วนใหญ่โปรแกรม assembler ของไมโครคอนโทรลเลอร์แต่ละตระกูลจะเป็นโปรแกรมที่ให้ใช้งานได้ฟรี ซึ่งแตกต่างจากการพัฒนาด้วยภาษาอื่น ๆ ที่อาจจะมีค่าใช้จ่าย</p> <p>การเขียนโปรแกรมภาษาแอสเซมบลีประกอบด้วย 4 ส่วนหลัก ซึ่งแยกได้โดยใช้ปุ่ม Tab คือ</p> <ol style="list-style-type: none">1. ลาเบล (Label) ใช้ในการอ้างถึงบรรทัดใดบรรทัดหนึ่งของโปรแกรมที่ทำการเขียนขึ้น โดยลาเบลจะต้องเขียนตามหลังด้วยเครื่องหมายโคลอน “:”2. รหัสนิมิก (Mnemonic) เป็นส่วนแสดงคำสั่งของไมโครคอนโทรลเลอร์ที่ต้องการให้กระทำ3. โอเพอเรนด์ (Operand) เป็นส่วนที่แสดงถึงตัวกระทำหรือถูกกระทำและข้อมูลที่ใช้ในการกระทำตามคำสั่งที่กำหนดโดยรหัสนิมิกก่อนหน้านี้4. คอมเมนต์ (Comment) เป็นส่วนที่ผู้เขียนโปรแกรมเขียนขึ้นเพื่อใช้ในการอธิบายคำสั่งที่กระทำ หรือผลของการกระทำคำสั่งในบรรทัดหรือโปรแกรมนั้น ๆ โดยคอมเมนต์จะต้องเขียนตามหลังด้วยเครื่องหมายเซมิโคลอน “;” และการเขียนภาษาแอสเซมบลีในแต่ละบรรทัดจะมีหรือไม่มีในส่วนคอมเมนต์ก็ได้ <p>ตารางที่ 1.3 แสดงตัวอย่างการแบ่งพื้นที่หน้ากระดาษในการเขียนโปรแกรมภาษาแอสเซมบลี</p>																				
<table><tr><th>Label</th><th>Mnemonic</th><th>Operand</th><th>Comment</th></tr><tr><td rowspan="4">Start:</td><td>ORG</td><td>0000H</td><td></td></tr><tr><td>MOV</td><td>SP,#128-32</td><td>;Set Stack Address</td></tr><tr><td>MOV</td><td>R2,#20</td><td>;R2 = 14H</td></tr><tr><td>END</td><td></td><td></td></tr></table>				Label	Mnemonic	Operand	Comment	Start:	ORG	0000H		MOV	SP,#128-32	;Set Stack Address	MOV	R2,#20	;R2 = 14H	END		
Label	Mnemonic	Operand	Comment																	
Start:	ORG	0000H																		
	MOV	SP,#128-32	;Set Stack Address																	
	MOV	R2,#20	;R2 = 14H																	
	END																			
<p>จากตารางที่ 1.3 ตัวอย่างการเขียนโปรแกรมภาษาแอสเซมบลี จะมีการใช้ชุดคำสั่งภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ของตระกูลนั้น ๆ และคำสั่งภาษาแอสเซมบลีเทียม ซึ่งเป็นชุดคำสั่งภาษาแอสเซมบลีของ Text Editor ที่เรารู้จักใช้งาน เพื่อทำการเขียนโปรแกรมภาษาแอสเซมบลี และตัวแอสเซมเบลอร์ของไมโครคอนโทรลเลอร์ตระกูลนั้น ๆ สามารถเข้าใจได้ และเมื่อทำการแอสเซมเบลอร์จะไม่เกิด Error ขึ้น ในตารางที่ 1.3 ตัวอย่างคำสั่งภาษาแอสเซมบลีเทียมได้แก่คำสั่ง ORG 0000H และ END เป็นต้น</p>																				

	ใบเนื้อหา	หน้าที่ 22
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

3. การใช้งานซอฟต์แวร์เพื่อเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์

3.1 การใช้งานโปรแกรม Keil uVision3 สำหรับเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล MCS51

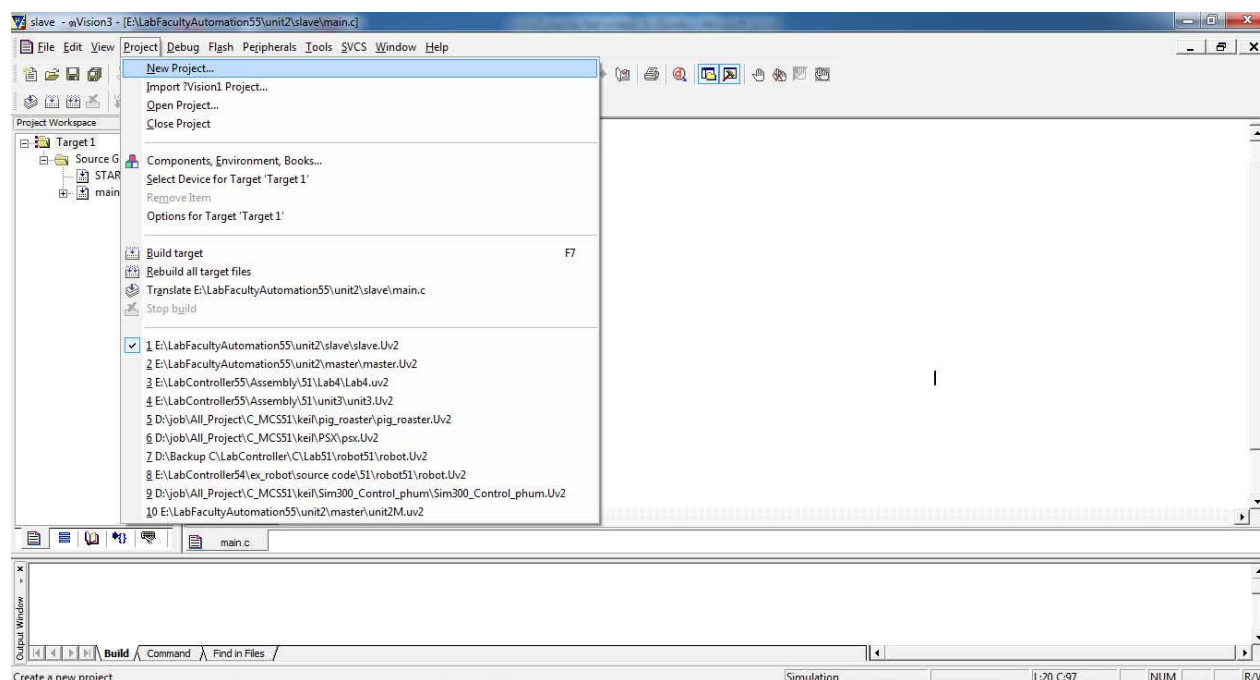
การใช้งานโปรแกรม Keil uVision3 สำหรับเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล MCS51 นั้นก่อนอื่นให้ทำการ Download Program Keil uVision3 แบบ Evaluation และทำการติดตั้ง ซึ่งจะมีข้อกำหนด คือสามารถ compile หรือ assembler ได้ไฟล์ .hex สูงสุดไม่เกิน 2Kbyte มีขั้นตอนดังนี้

1. ให้ทำการดับเบิลคลิกที่ไอคอนโปรแกรม Keil uVision3 ด้านหน้า Desktop ตามรูปที่ 1.28




รูปที่ 1.28 รูปแสดงไอคอนของโปรแกรม Keil uVision3

2. คลิกที่เมนู Project > New Project ตามรูปที่ 1.29

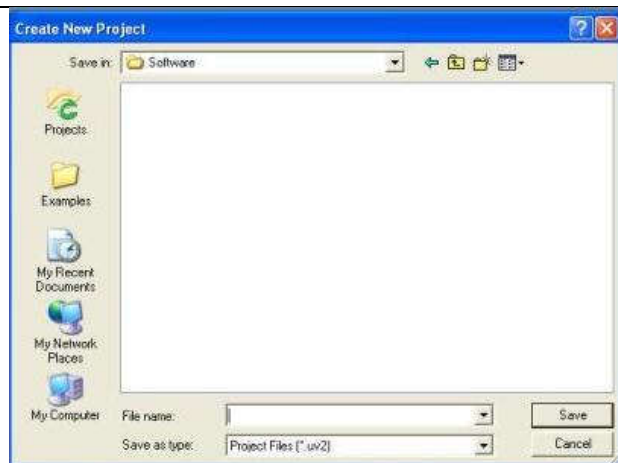


รูปที่ 1.29 รูปแสดงโปรแกรม Keil uVision3

3. เมื่อปรากฏรูปตามรูปที่ 1.30 ให้ทำการเลือก Drive และ Folder ที่ต้องการจะทำการเก็บโปรเจกต์ไฟล์ที่กำลังจะสร้างขึ้น จากนั้นให้ทำการตั้งชื่อโปรเจกต์ไฟล์ในช่อง File name เมื่อตั้งชื่อไฟล์เรียบร้อยแล้วให้ทำการคลิกที่ปุ่ม Save

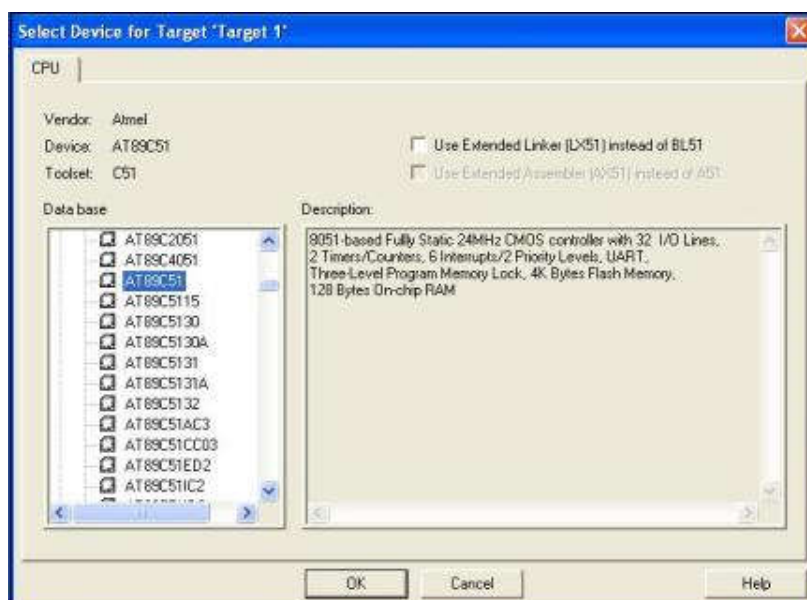
	ใบเนื้อหา	หน้าที่ 23
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.30 รูปแสดงหน้าต่าง Create New Project

4. หลังจากที่ยืนยันโปรเจกต์แล้ว โปรแกรม uVision 3 จะแสดงหน้าต่าง Select Device for Target 'Target1' ขึ้นมาตามรูปที่ 1.31 เพื่อให้เราเลือกชิพที่จะใช้งานจาก Device Database โดยให้เลือกชิพที่ต้องการซึ่งในตัวอย่างเลือกชิพ AT89C51 ของบริษัท ATMEL

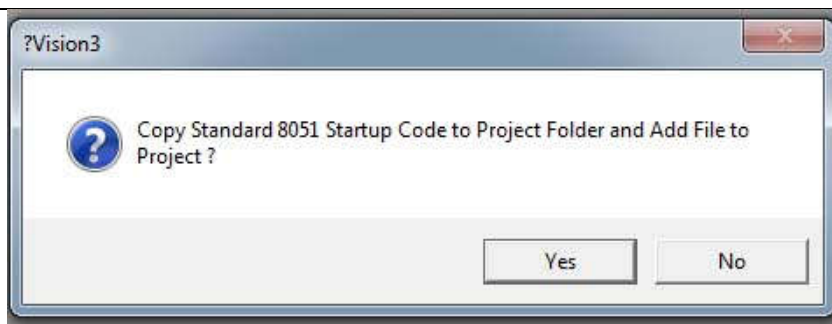


รูปที่ 1.31 รูปแสดงหน้าต่าง Select Device for Target 'Target1'

5. เมื่อทำการเลือกเบอร์ CPU เรียบร้อยแล้วจะปรากฏหน้าต่างตามรูปที่ 1.32 ซึ่งถ้าเขียนโปรแกรมภาษาแอสเซมบลีให้เลือกตอบ No แต่ถ้าเขียนโปรแกรมภาษาซีให้ตอบ Yes

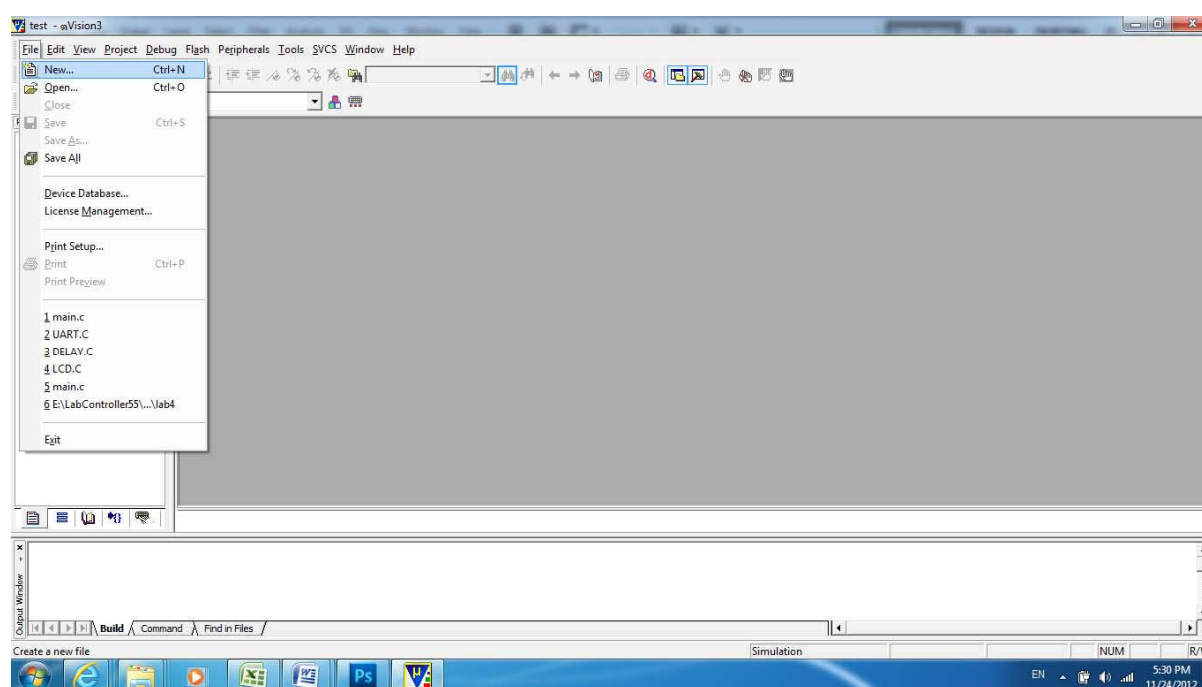
	ใบเนื้อหา		หน้าที่ 24
	ชื่อวิชา	ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.32 รูปแสดงหน้าต่างให้เลือกสร้างไฟล์ 8051 Startup Code

6. เมื่อเข้าสู่โปรแกรมให้ทำการเลือกเมนู File > new ตามรูปที่ 1.33

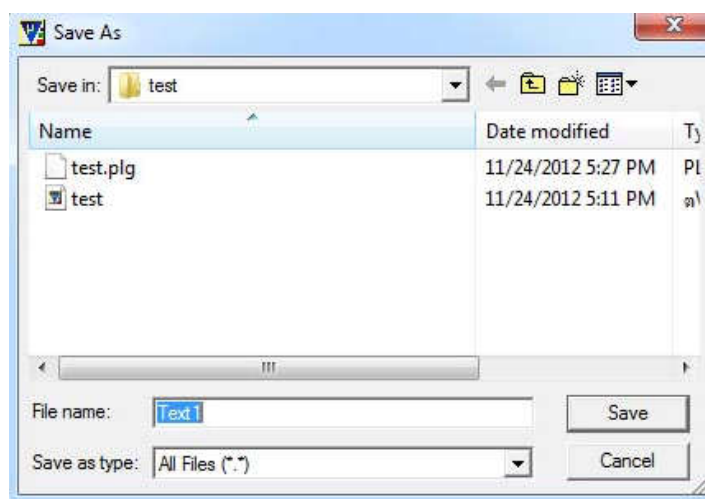


รูปที่ 1.33 รูปแสดงขั้นตอนสร้าง new file

7. ให้ทำการขยายหน้าจอ work sheet แล้วทำการเลือกเมนู File > Save จะปรากฏหน้าต่าง Save as ตามรูปที่ 1.34 ให้ทำการตั้งชื่อไฟล์ในช่อง File name โดยให้ต่อท้ายด้วยนามสกุล .asm

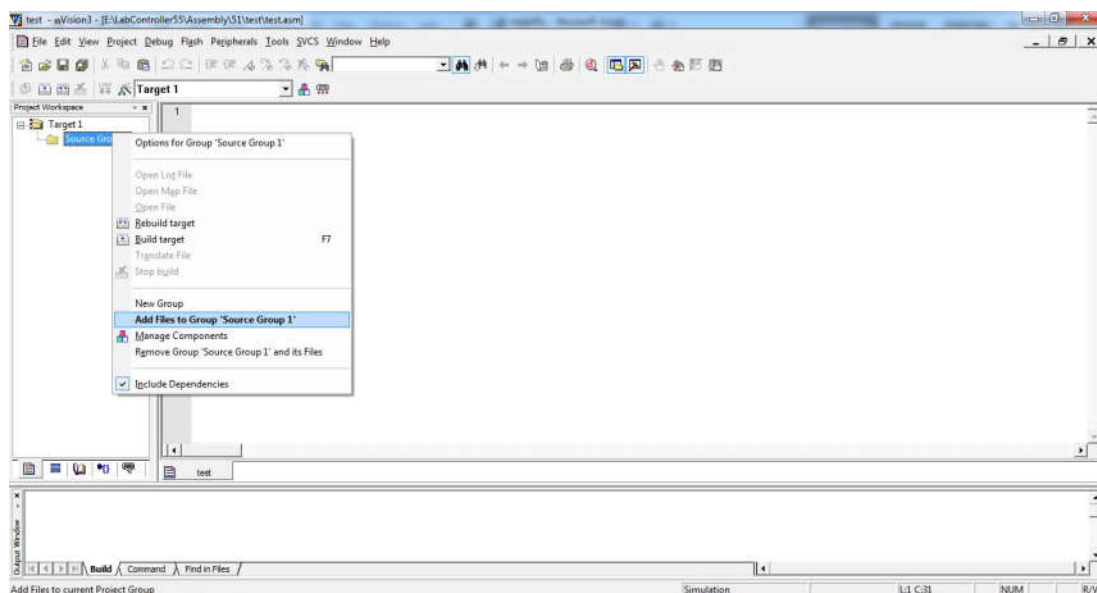
	ใบเนื้อหา		หน้าที่ 25
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.34 แสดงรูปหน้าต่างการตั้งชื่อไฟล์นามสกุล .asm

8. บริเวณหน้าต่างโปรแกรมทางด้านซ้ายมือให้ทำการคลิกที่เครื่องหมายถูกหน้าคำว่า Target1 แล้วทำการคลิกขวาที่คำว่า Source Group 1 จะปรากฏหน้าต่างดังรูปที่ 1.35 ให้ทำการเลือก Add File to Group “Source Group 1”

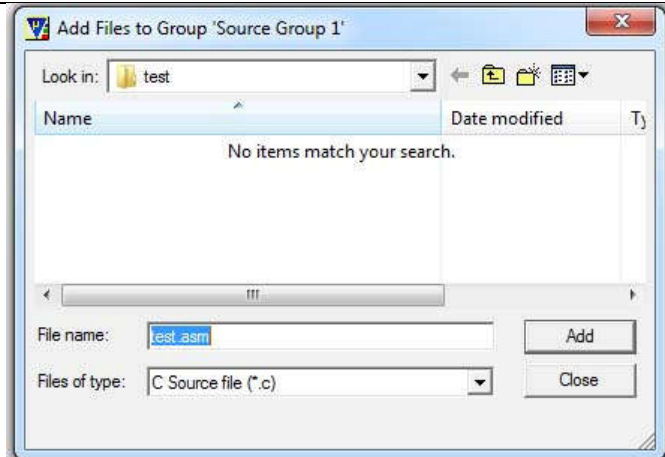


รูปที่ 1.35 แสดงหน้าต่างการเลือกการเลือก Add File to Group “Source Group 1”

9. เมื่อปรากฏหน้าต่าง Add File to Group “Source Group 1” ตามรูปที่ 1.36 ในช่อง File name ให้พิมพ์ชื่อไฟล์พร้อมนามสกุล .asm ที่ได้ทำการบันทึกไว้แล้ว หลังจากนั้นให้ทำการคลิกที่ปุ่ม Add ก็จะปรากฏไฟล์ที่ได้ทำการเลือกไว้ปรากฏที่หน้าต่างโปรแกรมทางด้านซ้ายมือซึ่งจะอยู่ในซบของ Source Group 1 โดยขั้นตอนนี้ให้ทำเพียงครั้งเดียวในการเขียนโปรแกรมหนึ่งโปรแกรม

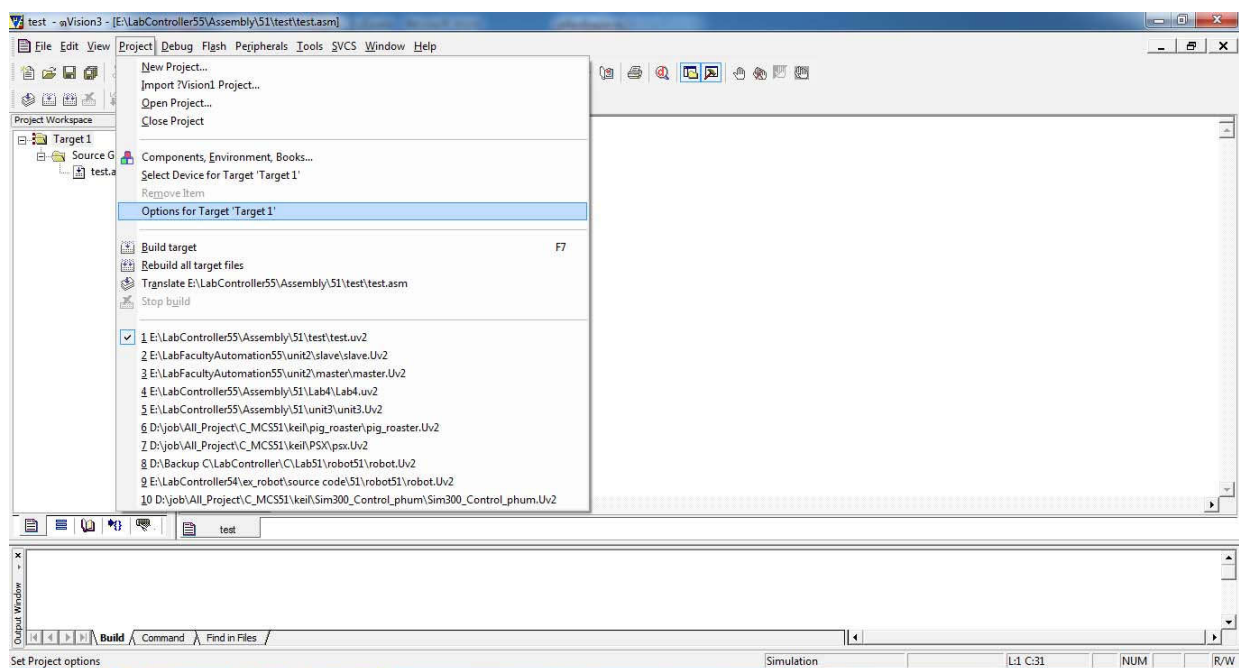
	ใบเนื้อหา		หน้าที่ 26
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.36 แสดงการเพิ่มไฟล์นามสกุล .asm เข้าสู่ Source Group 1

10. ให้ทำการเลือกเมนู Project > Options for Target 'Target 1' ดังรูปที่ 1.37

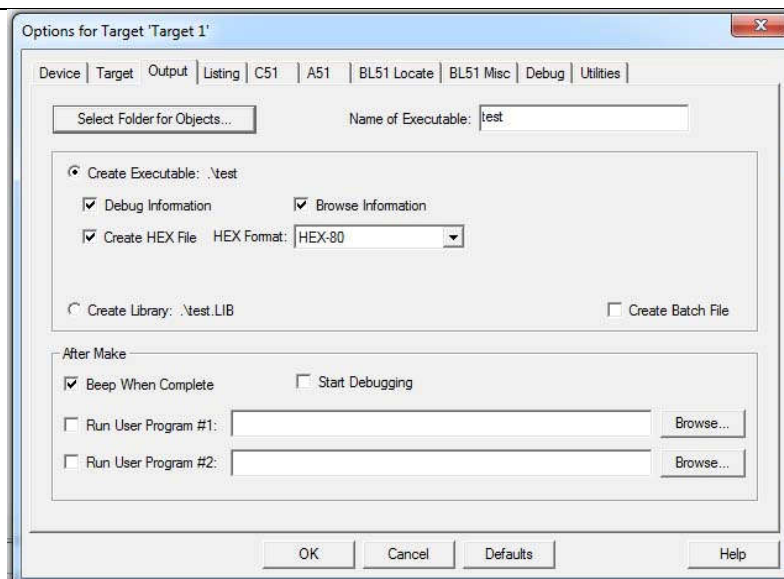


รูปที่ 1.37 แสดงขั้นตอนการเลือกเมนู Options for Target 'Target 1'

11. จากข้อ 10 จะปรากฏหน้าต่าง Options for Target 'Target1' ดังรูปที่ 1.38 ให้ทำการเลือกแท็บ Output แล้วทำการคลิกตรงช่อง Create Hex File ให้มีเครื่องหมายถูก จากนั้นให้คลิกที่ปุ่ม Ok โดยขั้นตอนนี้จะกระทำเพียงครั้งเดียวต่อ 1 โปรแกรม

	ใบเนื้อหา		หน้าที่ 27
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์



รูปที่ 1.38 รูปแสดงการเลือกเมนู Create HEX File

12. เมื่อทำขั้นตอน 1 – 11 เรียบร้อยแล้วให้ทำการเขียนโปรแกรมภาษาแอสเซมบลีบริเวณ Work Sheet เมื่อทำการเขียนโปรแกรมเรียบร้อยแล้วให้ทำการ Assembler โปรแกรมโดยเข้าเมนู Project > Build target แล้วให้สังเกตบริเวณหน้าต่างโปรแกรมด้านล่างว่ามีข้อความ 0 Error หรือไม่ ถ้าไม่ให้ทำการแก้ไขโปรแกรมแล้วทำการ Build target ใหม่จนกว่าหน้าต่าง Output จะปรากฏข้อความ 0 Error


3.2 การใช้งานโปรแกรม MPLAB X สำหรับเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล PIC16F

การใช้งานโปรแกรม MPLAB X สำหรับเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล PIC16F นั้นก่อนอื่นให้ทำการ Download Program MPLAB X V4.2 หรือสูงกว่าและทำการติดตั้ง หลังจากนั้นทำการ Download Program XC8 แล้วทำการติดตั้งเพื่อเตรียมการสำหรับเขียนโปรแกรมภาษาซี

1. ให้ทำการดับเบิลคลิกที่ไอคอนโปรแกรม MPLAB X IDE ด้านหน้า Desktop ตามรูปที่ 1.39

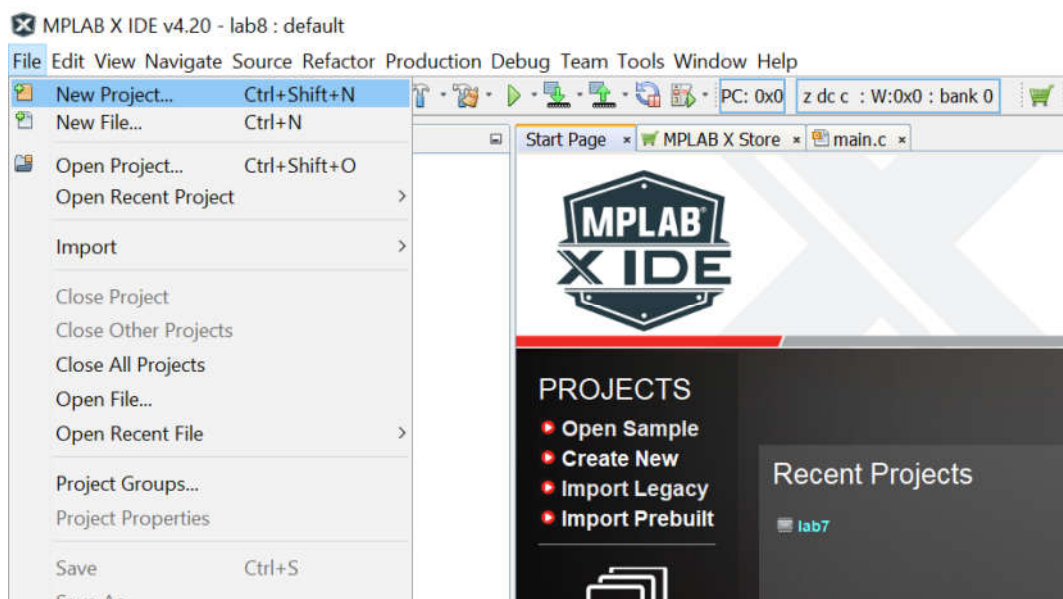


รูปที่ 1.39 แสดงรูปไอคอนของโปรแกรม MPLAB X IDE V4.20

	ใบเนื้อหา		หน้าที่ 28
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

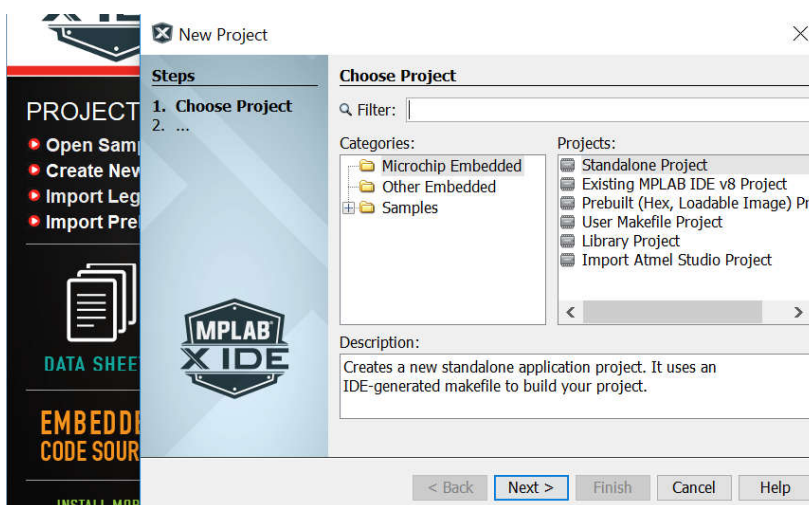
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

2. คลิกที่เมนู File > New Project ตามรูปที่ 1.40




รูปที่ 1.40 แสดงวิธีการสร้าง New project

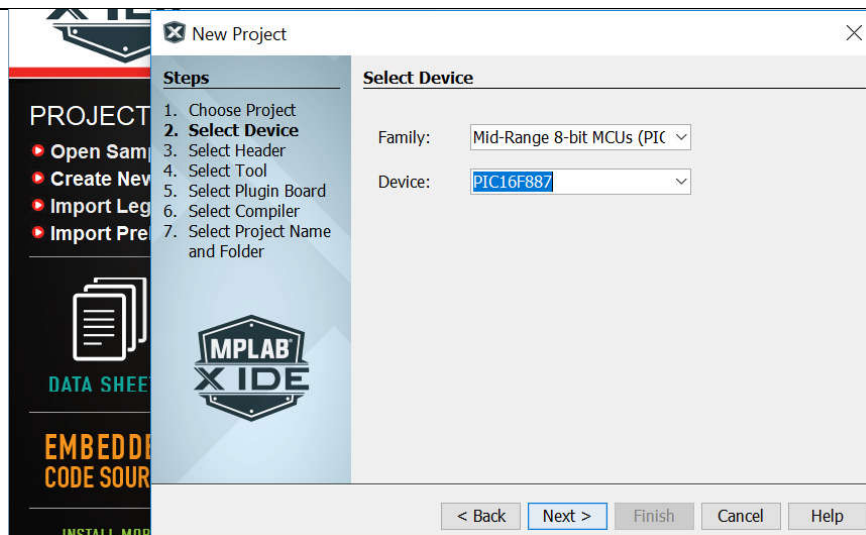
3. เมื่อปรากฏหน้าต่าง New project Choose Project ตามรูปที่ 1.41 ให้คลิกที่ปุ่ม Next หลังจากนั้นจะปรากฏหน้าต่างให้เลือก Select Device ให้ทำการเลือกไมโครคอนโทรลเลอร์บอร์ด PIC16F887 ตามรูปที่ 1.42 แล้วคลิกปุ่ม Next



รูปที่ 1.41 แสดงหน้าต่าง New project Choose Project

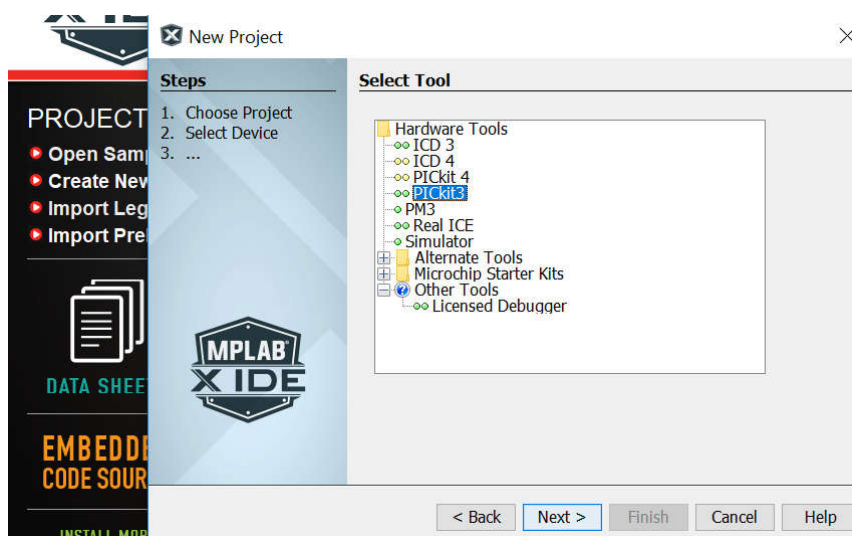
	ใบเนื้อหา	หน้าที่ 29
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.42 แสดงหน้าต่าง Select Device

4. จากข้อ 3 เมื่อคลิกที่ปุ่ม Next แล้วจะปรากฏหน้าต่าง Select Tool ให้ทำการเลือก PICkit3 แล้วทำการคลิกที่ปุ่ม Next ดังรูปที่ 1.43

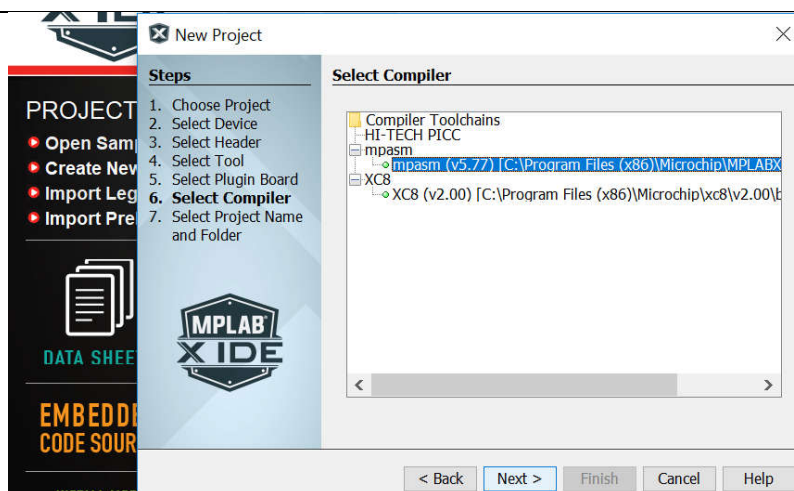


รูปที่ 1.43 แสดงหน้าต่าง Select Tool

5. จากข้อ 4 เมื่อคลิกที่ปุ่ม Next แล้วจะปรากฏหน้าต่าง Select Compiler ให้ทำการเลือก mpasm แล้วทำการคลิกที่ปุ่ม Next ดังรูปที่ 1.44

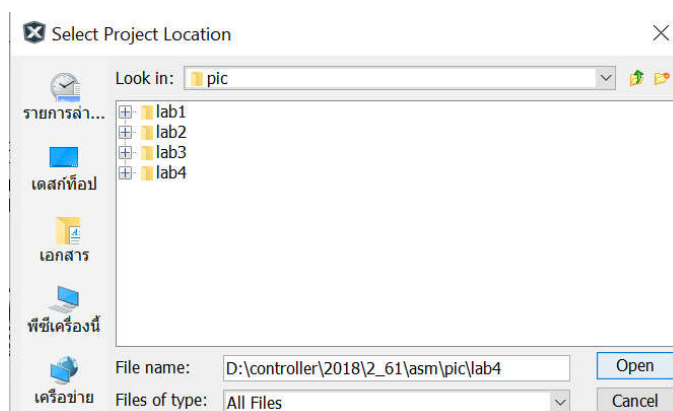
	ใบเนื้อหา		หน้าที่ 30
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.44 หน้าต่างแสดง Select Compiler

6. จากข้อ 5 เมื่อคลิกที่ปุ่ม Next แล้วจะปรากฏหน้าต่าง Select Project Name and Folder ซึ่งในช่อง Project Location ให้ทำการคลิกที่ปุ่ม Browse แล้วจะปรากฏหน้าต่าง Select Project Location ดังรูปที่ 1.45 ให้ทำการเลือก Drive เลือก Folder ที่ต้องการจะบันทึก Project File ซึ่งในช่อง File name จะปรากฏ path ที่ต้องการจะบันทึก Project File แล้วคลิกปุ่ม Open เพื่อกลับเข้าหน้าต่าง Select Project Name and Folder

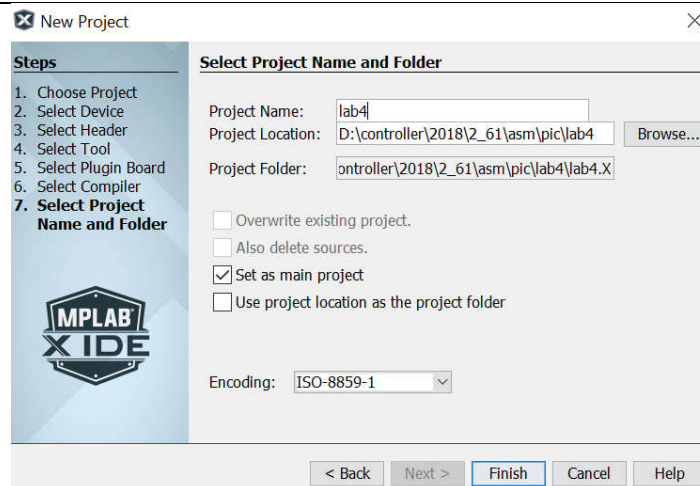


รูปที่ 1.45 แสดงหน้าต่าง Select Project Location

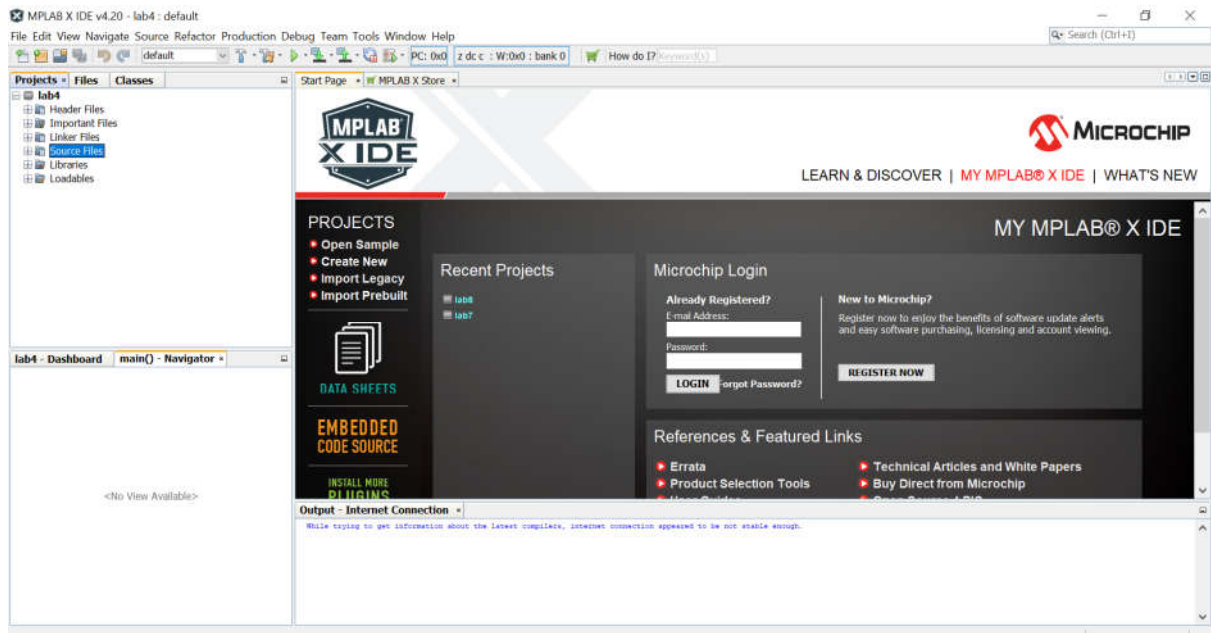
7. จากข้อ 6 เมื่อกลับเข้าสู่หน้าต่าง Select Project Name and Folder ในช่อง Project Name ให้ทำการพิมพ์ชื่อ Project Name ที่เราต้องการโดยที่ไม่ต้องใส่นามสกุลของไฟล์ แล้วให้คลิกที่ปุ่ม Finish เพื่อสิ้นสุดการสร้างโปรเจกต์ไฟล์ดังรูปที่ 1.46 และ 1.47

	ใบเนื้อหา		หน้าที่ 31
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.46 แสดงหน้าต่าง Select Project Name and Folder

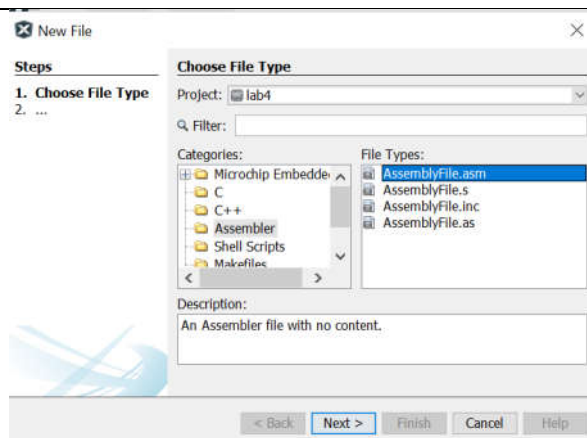


รูปที่ 1.47 แสดงการเตรียมการสร้างไฟล์ asm

8. เมื่อปรากฏหน้าจอโปรแกรม MPLAB X IDE ให้คลิกที่ข้อความ Source Files แล้วให้เข้าเมนู File > New File จะปรากฏหน้าต่าง New File ดังรูปที่ 1.48 แล้วให้ทำการเลือก Assembler ในช่อง Categories และเลือก AssemblyFile.asm ในช่อง File Types แล้วทำการคลิกที่ปุ่ม Next

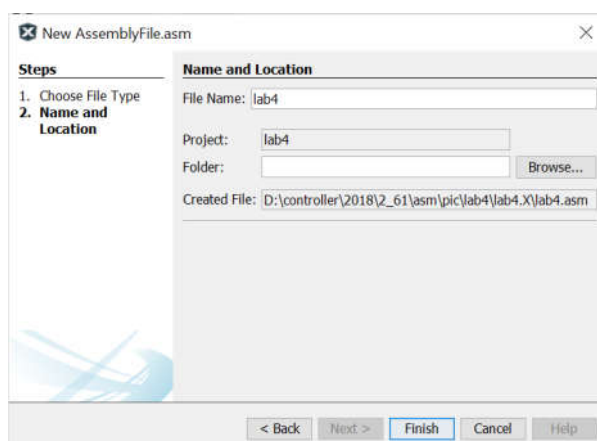
	ใบเนื้อหา		หน้าที่ 32
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.48 หน้าต่างแสดงให้เลือกชนิดของไฟล์ที่ต้องการสร้าง

9. จากข้อ 8 เมื่อปรากฏหน้าต่าง Name and Location ในช่อง File Name ให้ทำการพิมพ์ชื่อ File Name ที่เราต้องการโดยที่ไม่ต้องใส่นามสกุลของไฟล์ แล้วให้คลิกที่ปุ่ม Finish เพื่อสิ้นสุดการสร้างไฟล์ Assembly ดังรูปที่ 1.49

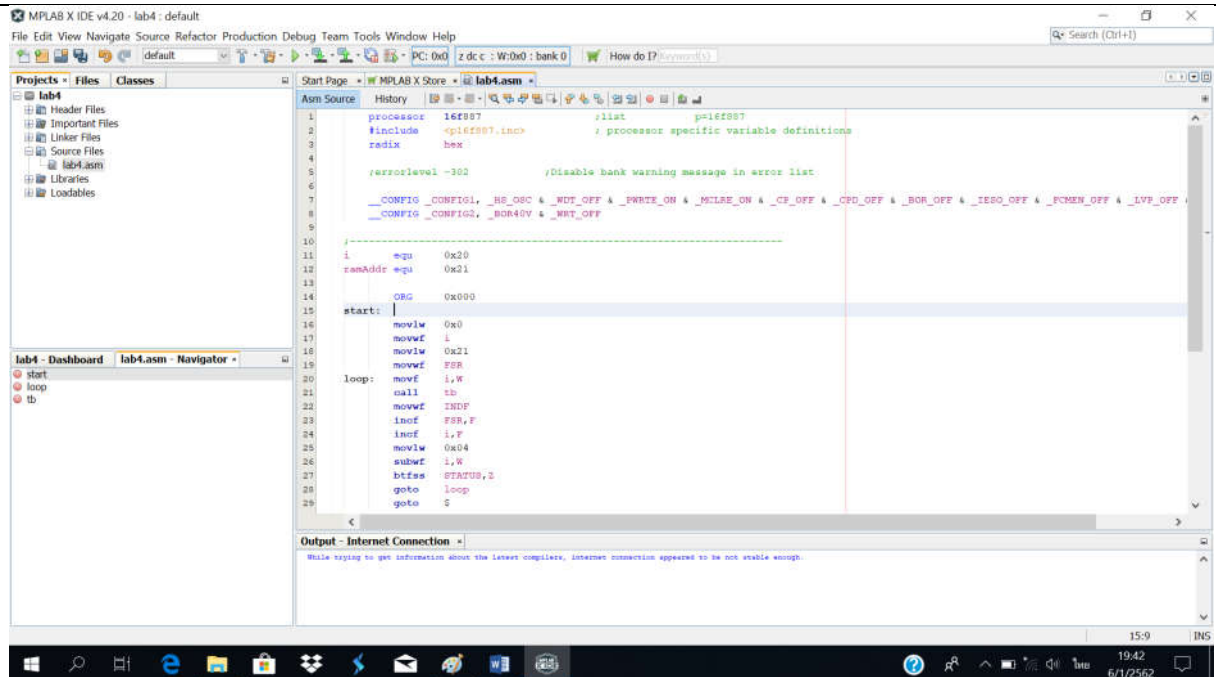


รูปที่ 1.49 รูปหน้าต่างแสดงให้ยอมรับการสร้างไฟล์นามสกุล .asm

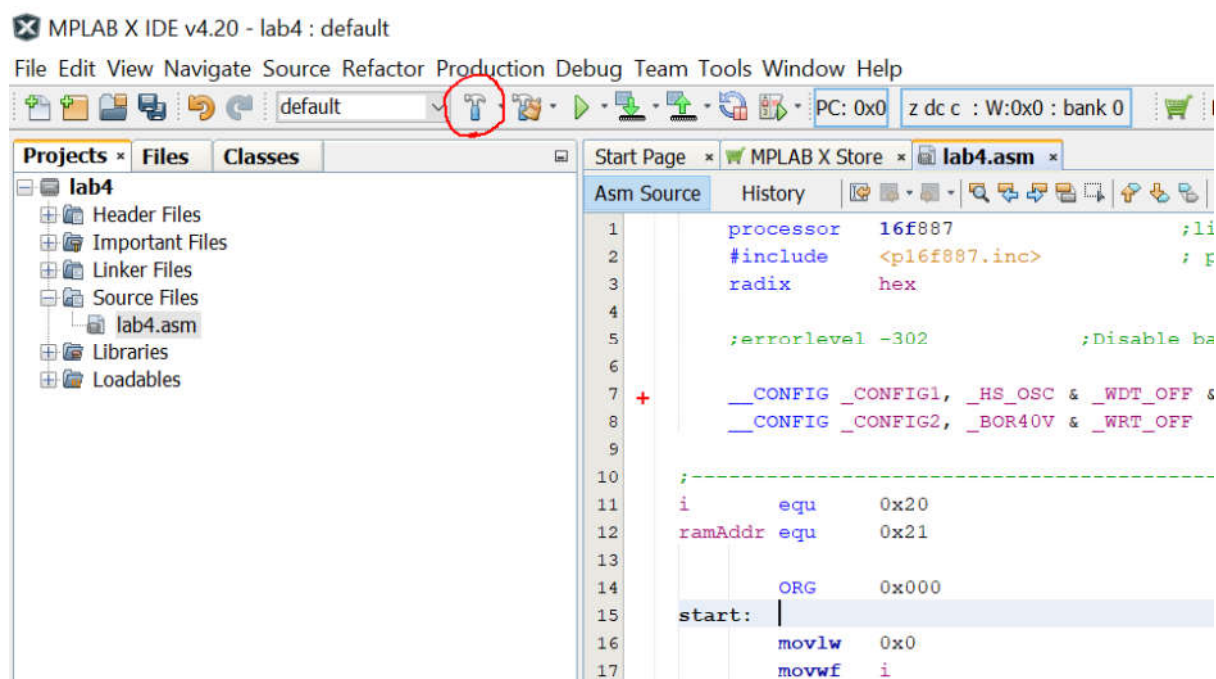
10. เมื่อกลับเข้าสู่โปรแกรม MPLAB X IDE ให้ทำการเขียนโปรแกรมภาษาแอสเซมบลีบริเวณ Work Sheet ดังรูปที่ 1.50 เมื่อทำการเขียนโปรแกรมเรียบร้อยแล้วให้ทำการ Assembler โปรแกรมโดยคลิกที่ไอคอนเมนู Build Main Project ดังรูปที่ 1.51 แล้วให้สังเกตบริเวณหน้าต่าง Output ของโปรแกรมด้านล่างว่ามีข้อความ Loading Completed หรือไม่ ถ้าไม่ให้ทำการแก้ไขโปรแกรมแล้วทำการ Build Main Project ใหม่จนกว่าหน้าต่าง Output จะปรากฏข้อความ Loading Completed ดังรูปที่ 1.52

	ใบเนื้อหา		หน้าที่ 33
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		


ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์



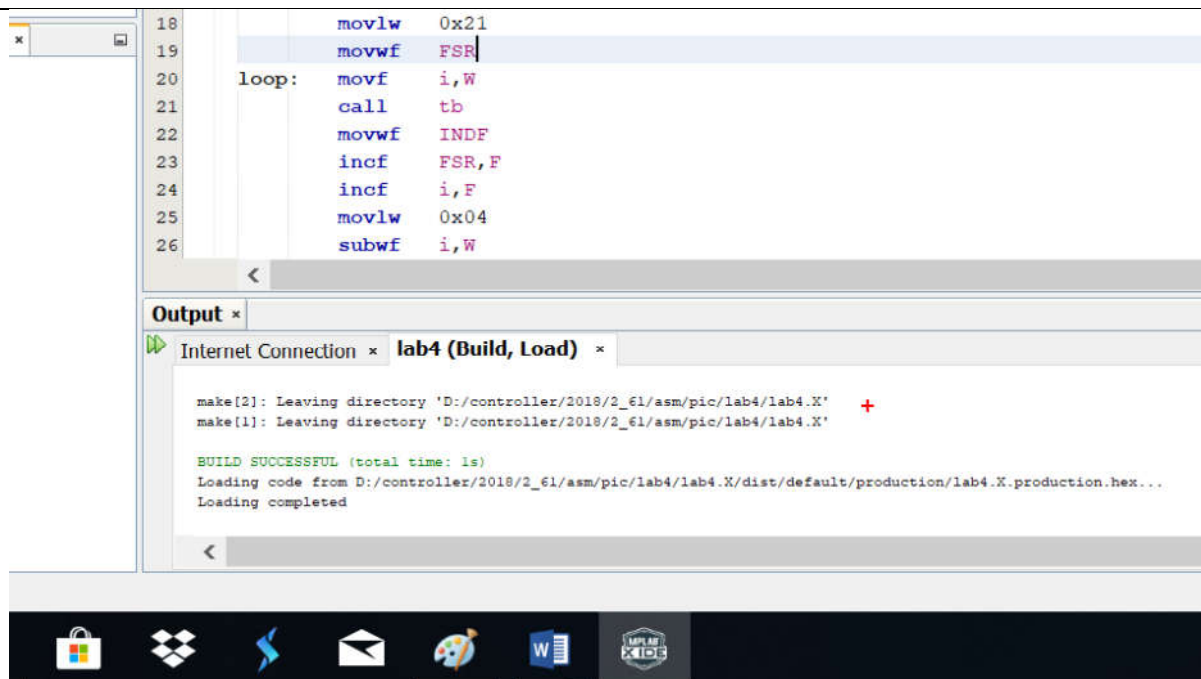
รูปที่ 1.50 รูปตัวอย่างการเขียนโปรแกรมแอสเซมบลีของ PIC16F



รูปที่ 1.51 รูปแสดงตำแหน่งเครื่องหมายการแอสเซมเบลอร์ของไมโครคอนโทรลเลอร์ PIC16F

	ใบเนื้อหา		หน้าที่ 34
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์



รูปที่ 1.52 แสดงผลการแอสเซมเบลอร์ของไมโครคอนโทรลเลอร์ PIC16F


3.3 การใช้งานโปรแกรม AVR Studio 6.2 สำหรับเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล AVR

การใช้งานโปรแกรม AVR Studio 6.2 สำหรับเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล AVR นั้นก่อนอื่นให้ทำการ Download Program AVR Studio 6.2 หรือสูงกว่าและทำการติดตั้ง หลังจากนั้นให้ทำตามขั้นตอนดังนี้

1. ให้ทำการดับเบิลคลิกที่ไอคอนโปรแกรม AVR Studio 6.2 ด้านหน้า Desktop ตามรูปที่ 1.53

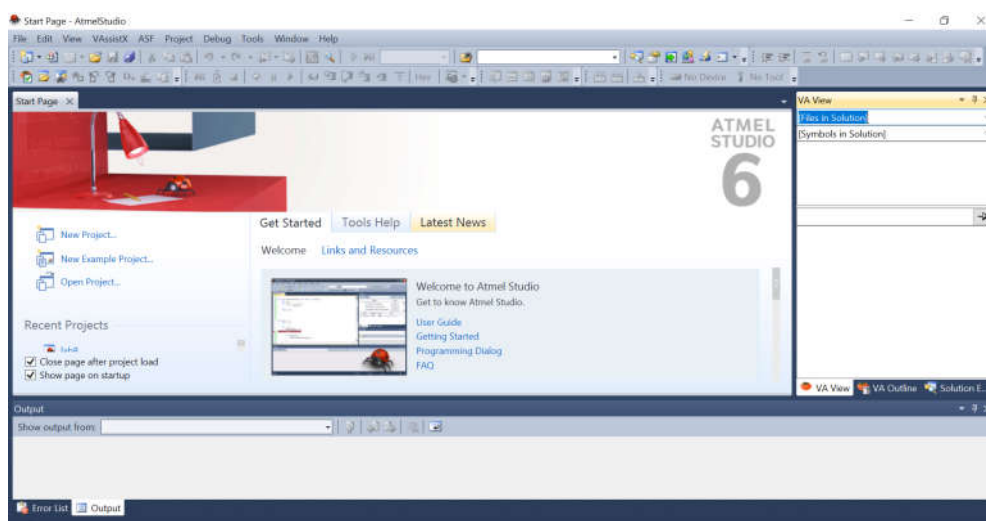


รูปที่ 1.53 รูปแสดงไอคอนของโปรแกรม AVR Studio 6.2

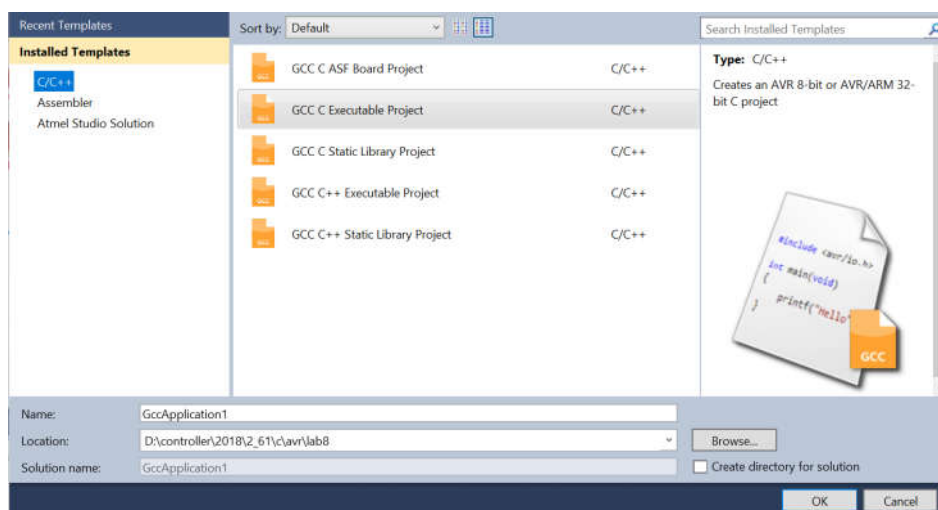
	ใบเนื้อหา	หน้าที่ 35
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

2. เมื่อปรากฏหน้าต่างโปรแกรมตามรูปที่ 1.54 ถ้าต้องการสร้างโปรเจกต์ใหม่ให้ทำการเลือกเมนู File > New > Project จะปรากฏหน้าต่าง New Project ตามรูปที่ 1.55




รูปที่ 1.54 รูปภาพแสดงหน้าต่างของโปรแกรม AVR Studio 6.2

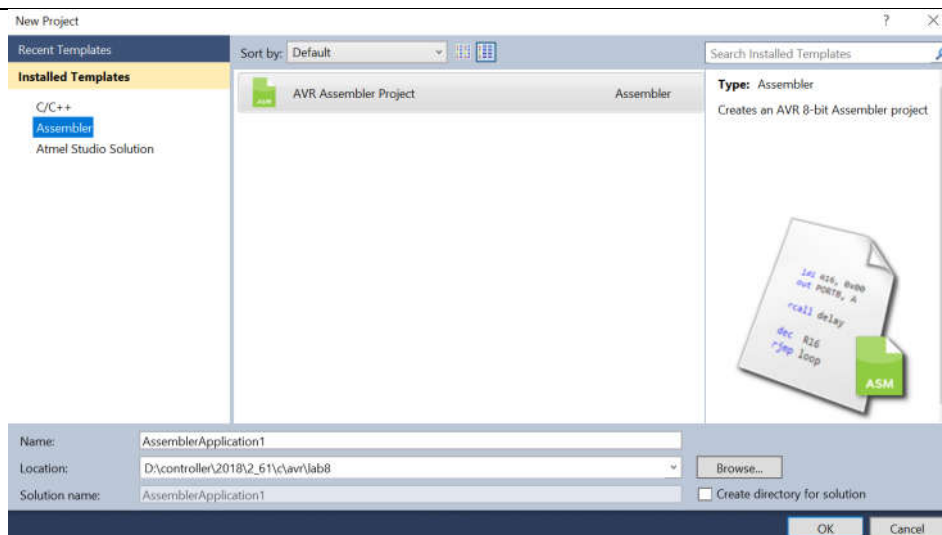


รูปที่ 1.55 แสดงรูปหน้าต่างของ New Project

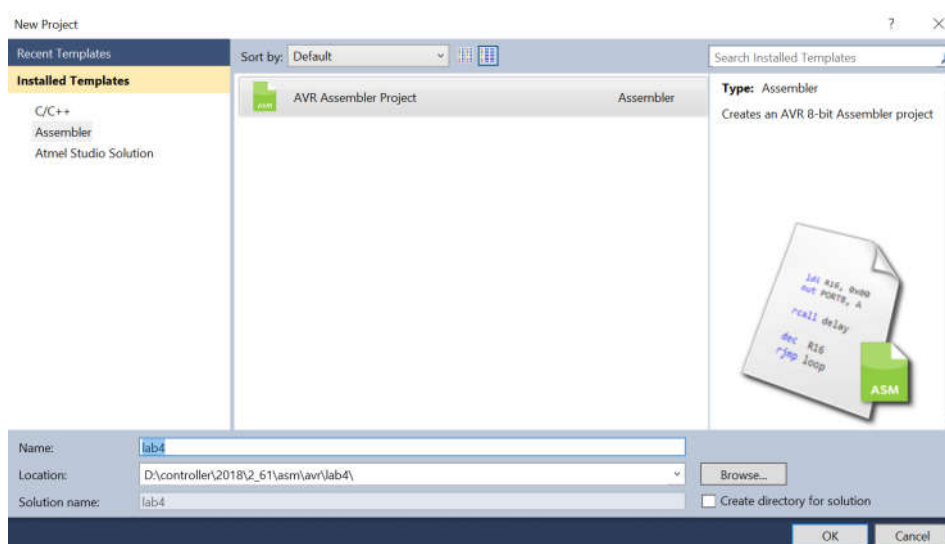
3. จากขั้นตอนที่ 2 เมื่อปรากฏหน้าต่าง New Project ตามรูปที่ 1.55 ให้ทำการเลือกข้อความ Assembler จะปรากฏหน้าต่างโปรแกรมตามรูปที่ 1.56 โดยในช่อง Location ให้ทำการเลือก Drive เลือก Folder ที่ต้องการเก็บโปรเจกต์ไฟล์ที่ต้องการสร้างขึ้นโดยการพิมพ์ path หรือคลิกที่ปุ่ม Browse แล้วทำการทำการเลือก Drive เลือก Folder ที่ต้องการเก็บโปรเจกต์ไฟล์ที่ต้องการสร้างขึ้น ส่วนในช่อง Name ให้ทำการพิมพ์ชื่อโปรเจกต์ที่ต้องการจะบันทึกโดยไม่ต้องใส่นามสกุลของไฟล์ดังรูปที่ 1.57 แล้วทำการคลิกปุ่ม OK

	ใบเนื้อหา		หน้าที่ 36
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.56 รูปเลือกเครื่องมือในการ compile เป็นแบบ Assembler

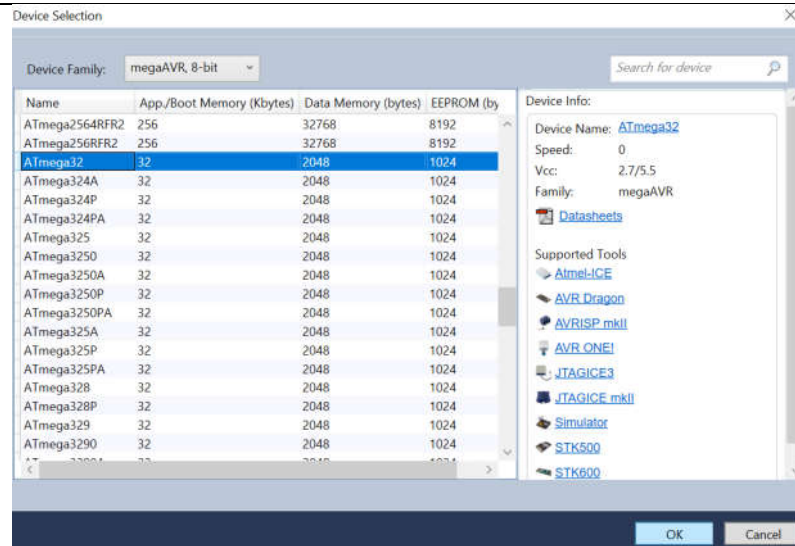


รูปที่ 1.57 รูปสร้างโปรเจคไฟล์

4. จากข้อ 3 เมื่อคลิกที่ปุ่ม Ok แล้วจะปรากฏหน้าต่าง Device Selection โดยในส่วนของ Device Family ให้เลือก megaAVR ,8bit แล้วให้ทำการคลิกเลือกไมโครคอนโทรลเลอร์เบอร์ Atmega32 แล้วทำการคลิกที่ปุ่ม OK ดังรูปที่ 1.58

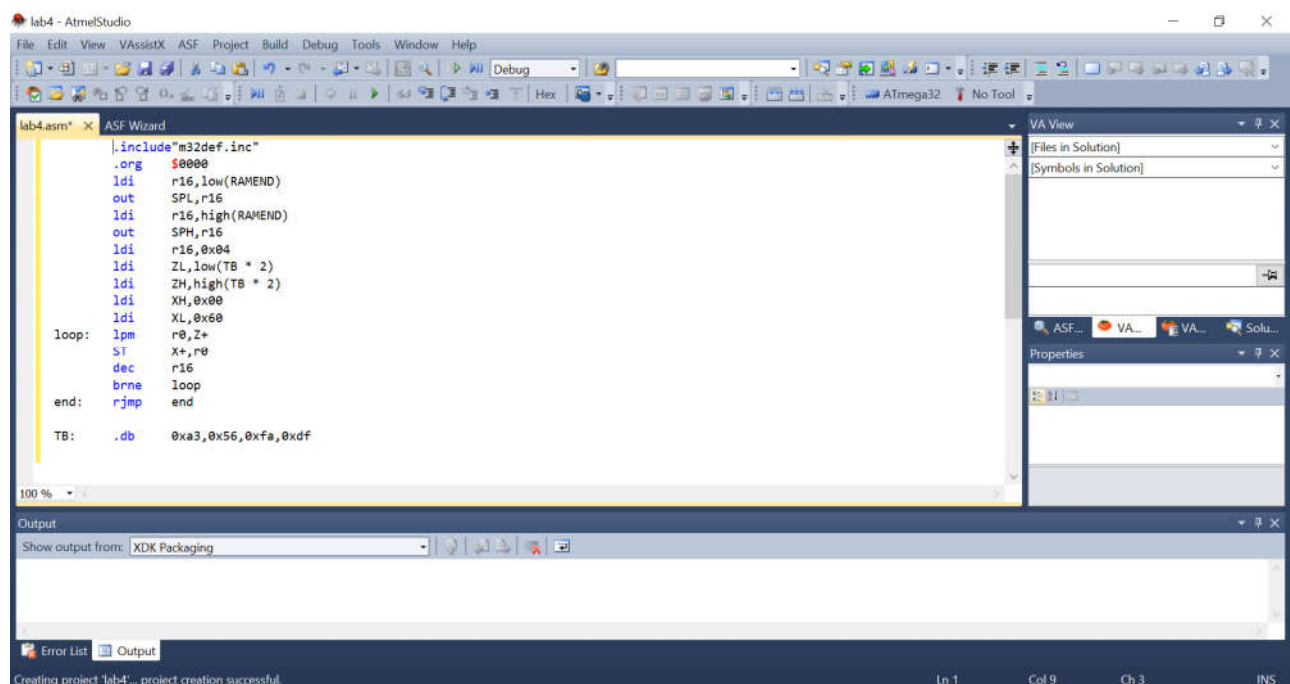
	ใบเนื้อหา	หน้าที่ 37
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.58 รูปหน้าต่างแสดงการเลือกเบอร์ไมโครคอนโทรลเลอร์

5. เมื่อเข้าสู่โปรแกรม AVR Studio 6.2 ในส่วนของ Work Sheet ให้ทำการเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR ให้เรียบร้อยดังรูปที่ 1.59

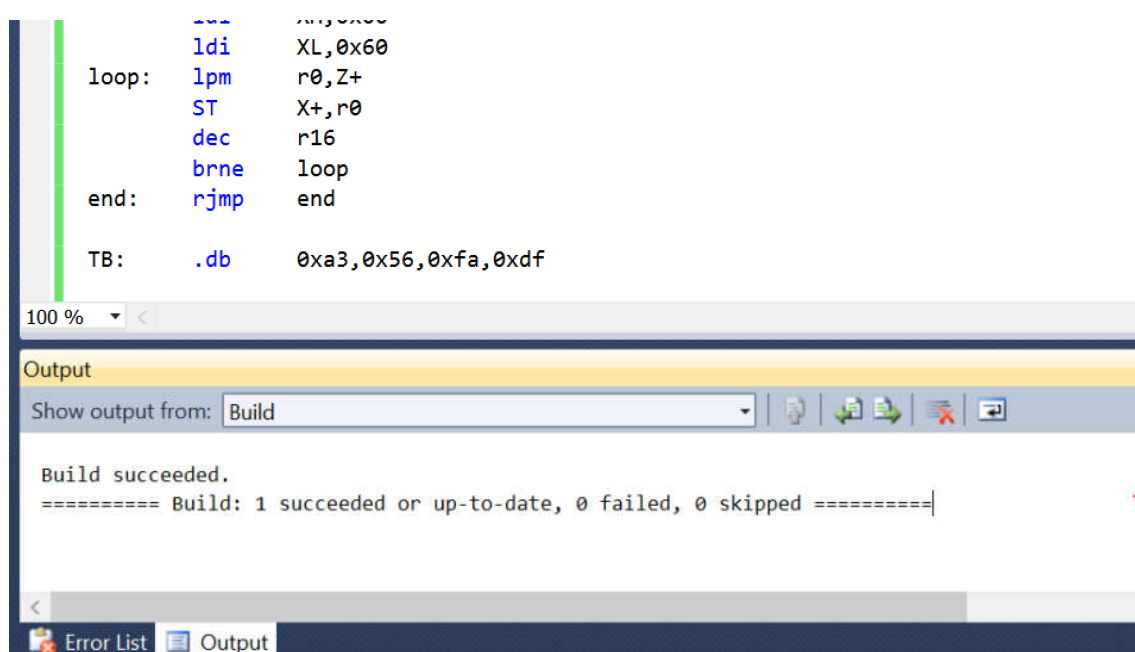


รูปที่ 1.59 รูปแสดงตัวอย่างการเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ AVR

	ใบเนื้อหา		หน้าที่ 38
	ชื่อวิชา	ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์


6. เมื่อทำขั้นตอนที่ 5 เรียบร้อยแล้วให้ทำการ Assembler โปรแกรมโดยเข้าเมนู Build > Build Solution แล้วให้สังเกตบริเวณหน้าต่าง Output ของโปรแกรมด้านล่างว่ามีข้อความ Build succeeded. หรือไม่ ถ้าไม่ให้ทำการแก้ไขโปรแกรมแล้วทำการ Build ใหม่จนกว่าหน้าต่าง Output จะปรากฏข้อความ Build succeeded. ดังรูปที่ 1.60



รูปที่ 1.60 รูปแสดงการ Assembler แล้วไม่เกิดข้อผิดพลาด

4. การใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์

การใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์มีขั้นตอนในการใช้งาน เช่นเดียวกับการใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรดิจิทัล เพียงแต่การใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์นั้น ก่อนที่จะทำการใช้งานโปรแกรมเราจะต้องเขียนโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์นั้น ๆ ให้สมบูรณ์ไม่เกิด Error เสียก่อน เพื่อที่จะได้ไฟล์นามสกุล .hex ของไมโครคอนโทรลเลอร์ตัวนั้น ๆ มาใช้งานเพื่อจำลองการทำงานต่อไป โดยการจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล MCS-51 ,PIC16F และ AVR มีขั้นตอนดังต่อไปนี้

	ใบเนื้อหา		หน้าที่ 39
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

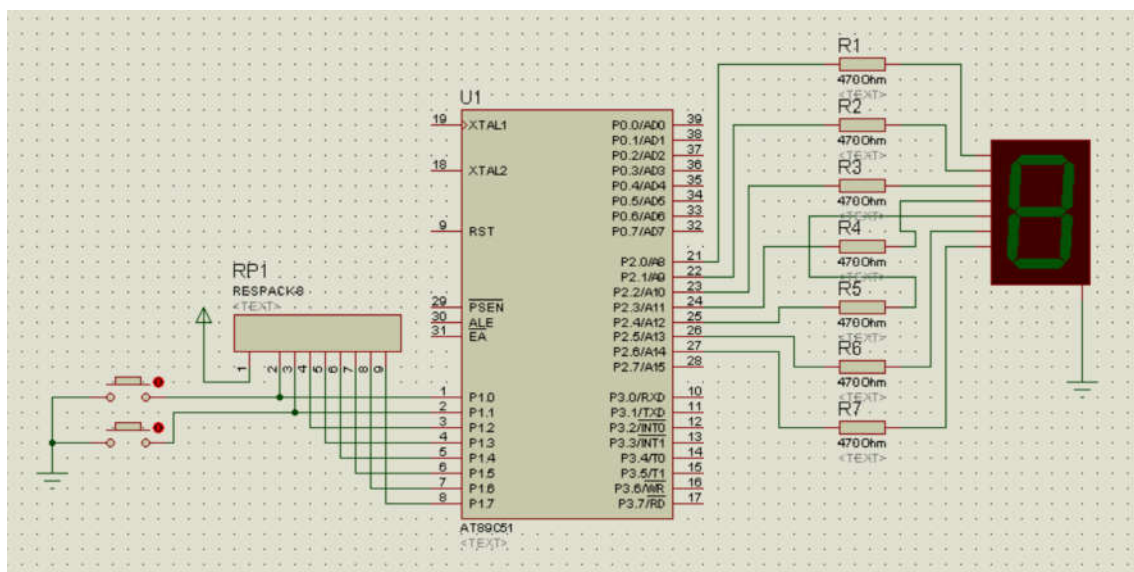
4.1 การใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล MCS51

1. ให้ทำการดับเบิลคลิกที่ไอคอนโปรแกรม Proteus ISIS ด้านหน้า Desktop ตามรูปที่ 1.61




รูปที่ 1.61 รูปไอคอน Proteus ISIS

2. ต่อวงจรเพื่อทดสอบการทำงานของไมโครคอนโทรลเลอร์ MCS-51 ตามใบงาน ดังรูปที่ 1.62

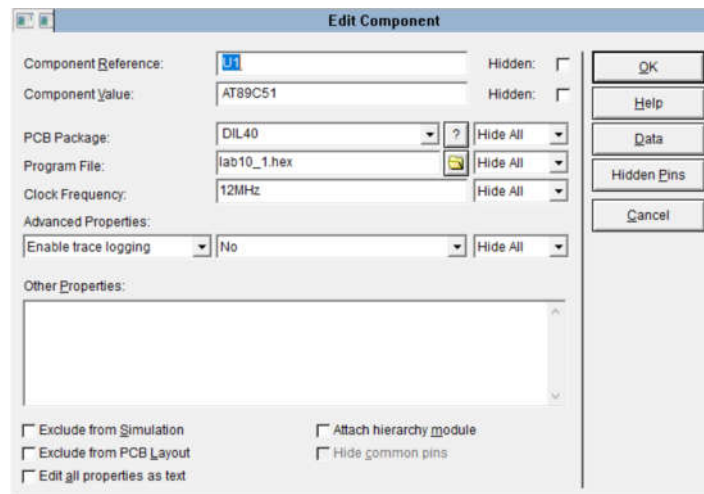


รูปที่ 1.62 ตัวอย่างการต่อวงจรเพื่อทดสอบการทำงานของไมโครคอนโทรลเลอร์ MCS-51

3. ดับเบิลคลิกที่ตัวอุปกรณ์ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C51 เพื่อให้โปรแกรมแสดงหน้าต่างการกำหนดคุณสมบัติของไอซีไมโครคอนโทรลเลอร์เบอร์ AT89C51 ตามรูปที่ 1.63

	ใบเนื้อหา		หน้าที่ 40
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

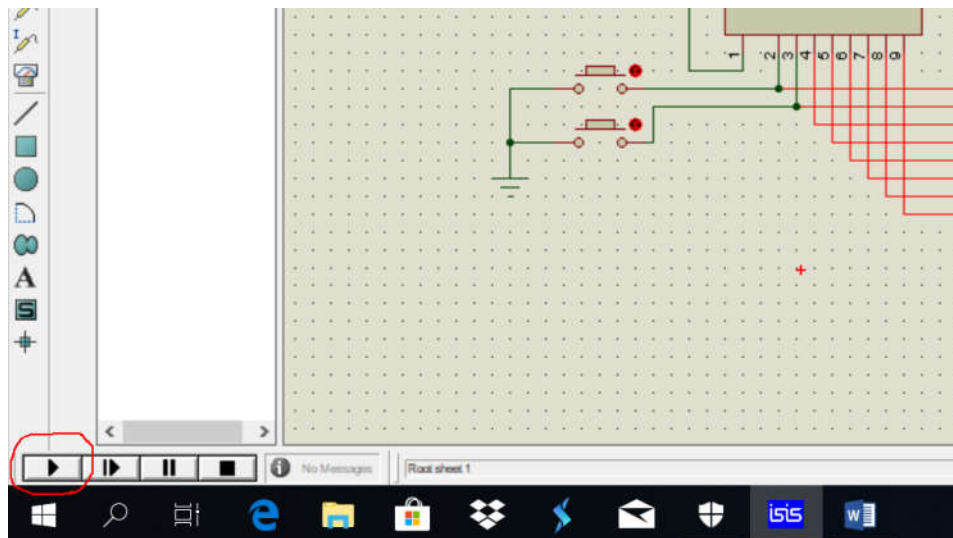
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.63 หน้าต่างการกำหนดคุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C51

4. จากรูปที่ 1.63 ในช่อง Program File ให้คลิกปุ่มไอคอนโฟลเดอร์เพื่อหาไฟล์โปรแกรมนามสกุล .hex ที่เราต้องการนำมา Simulate ของไมโครคอนโทรลเลอร์ AT89C51 ที่ได้จากการ Build ของโปรแกรม Keil uVision3 ส่วนในช่อง Clock Frequency ให้กำหนดค่าเป็น 12MHz หลังจากนั้นให้คลิกที่ปุ่ม OK

5. ทำการ Simulate วงจรโดยการคลิกที่ไอคอนตามรูปที่ 1.64



รูปที่ 1.64 แสดงตำแหน่งการคลิกปุ่ม play เพื่อเริ่มต้นการจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C51

	ใบเนื้อหา		หน้าที่ 41
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

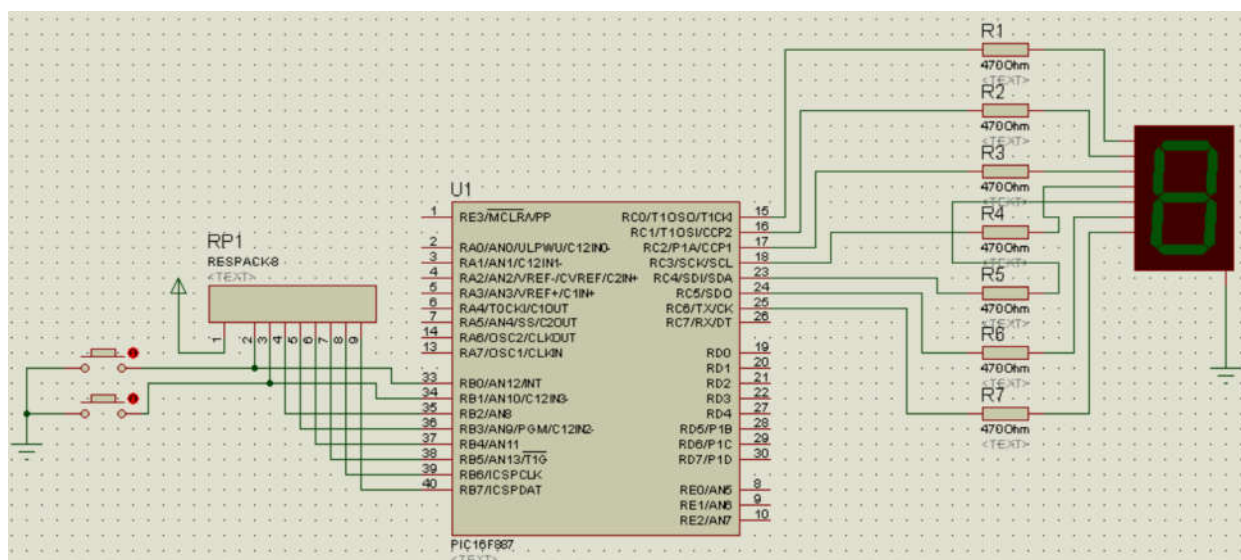
4.2 การใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล PIC16F

1. ให้ทำการดับเบิลคลิกที่ไอคอนโปรแกรม Proteus ISIS ตามรูปที่ 1.65




รูปที่ 1.65 รูปไอคอน Proteus ISIS

2. ต่อวงจรเพื่อทดสอบการทำงานของไมโครคอนโทรลเลอร์ PIC16F ตามใบงาน ดังรูปที่ 1.66

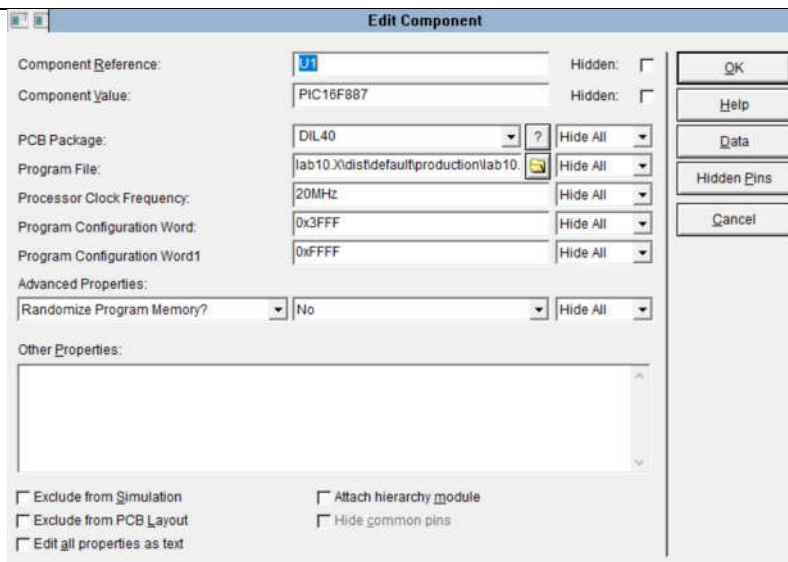


รูปที่ 1.66 ตัวอย่างการต่อวงจรเพื่อทดสอบการทำงานของไมโครคอนโทรลเลอร์ PIC16F

3. ดับเบิลคลิกที่ตัวอุปกรณ์ไมโครคอนโทรลเลอร์ตระกูล PIC16F เบอร์ PIC16F887 เพื่อให้โปรแกรมแสดงหน้าต่างการกำหนดคุณสมบัติของไอซีไมโครคอนโทรลเลอร์เบอร์ PIC16F887 ตามรูปที่ 1.67

	ใบเนื้อหา		หน้าที่ 42
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

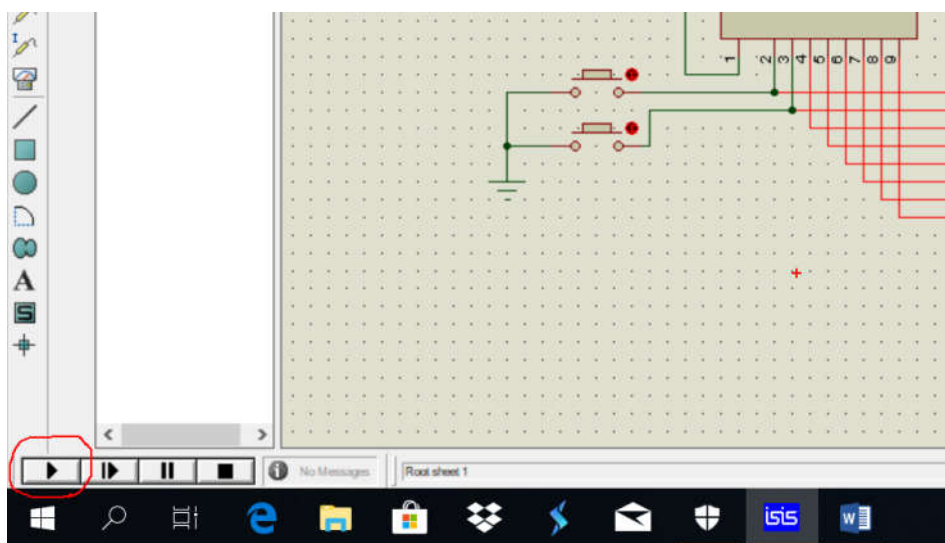
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.67 หน้าต่างการกำหนดคุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล PIC16F เบอร์ PIC16F887

4. จากรูปที่ 1.67 ในช่อง Program File ให้คลิกปุ่มไอคอนโฟลเดอร์เพื่อหาไฟล์โปรแกรมนามสกุล .hex ที่เราต้องการนำมา Simulate ของไมโครคอนโทรลเลอร์ PIC16F887 ที่ได้จากการ Build ของโปรแกรม MPLAB – X IDE ส่วนในช่อง Processor Clock Frequency ให้กำหนดค่าเป็น 20MHz หลังจากนั้นให้คลิกที่ปุ่ม OK

5. ทำการ Simulate วงจรโดยการคลิกที่ไอคอนตามรูปที่ 1.68



รูปที่ 1.68 แสดงตำแหน่งการคลิกปุ่ม play เพื่อเริ่มต้นการจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล PIC16F เบอร์ PIC16F887

	ใบเนื้อหา		หน้าที่ 43
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์

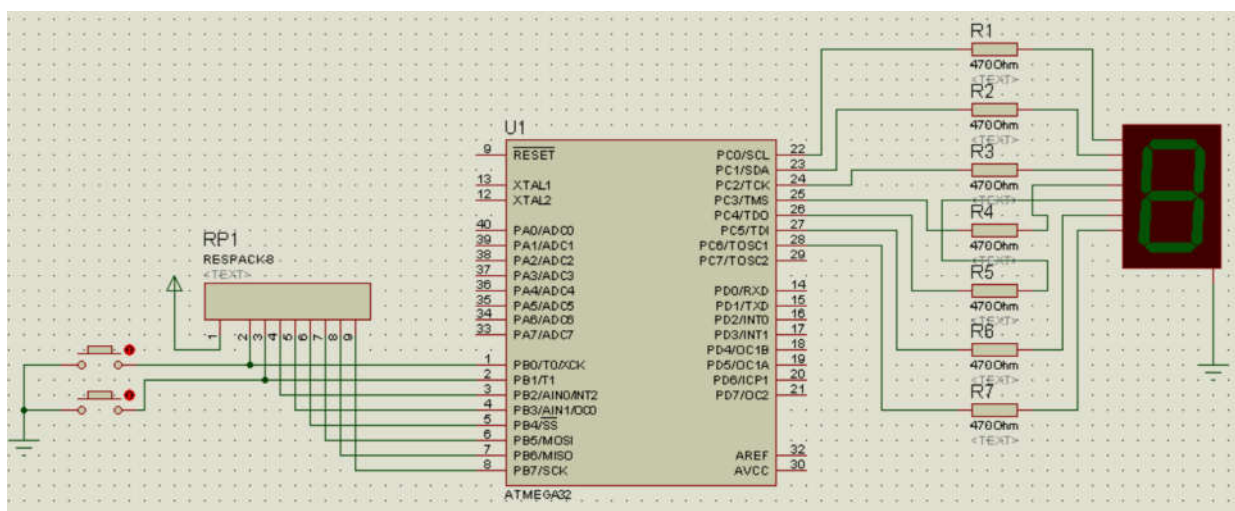
4.3 การใช้งานโปรแกรม Proteus เพื่อจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล AVR

1. ให้ทำการดับเบิลคลิกที่ไอคอนโปรแกรม Proteus ISIS ตามรูปที่ 1.69




รูปที่ 1.69 รูปไอคอน Proteus ISIS

2. ต่อวงจรเพื่อทดสอบการทำงานของไมโครคอนโทรลเลอร์ AVR ตามใบงาน ดังรูปที่ 1.70

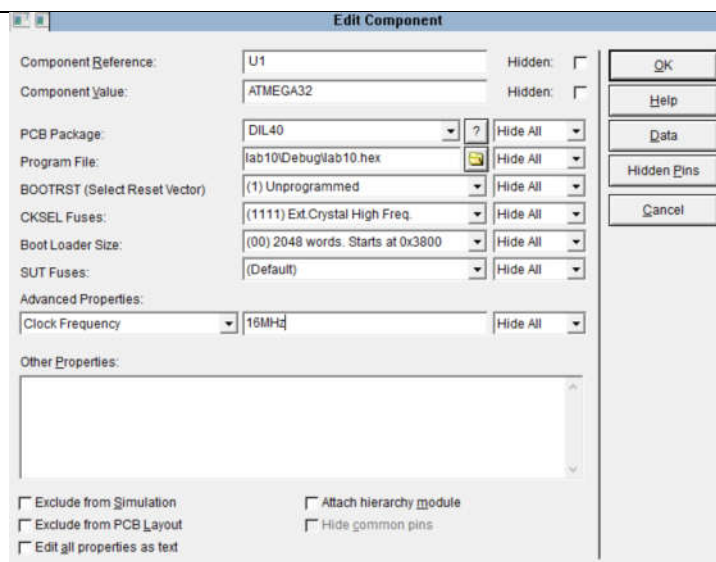


รูปที่ 1.70 ตัวอย่างการต่อวงจรเพื่อทดสอบการทำงานของไมโครคอนโทรลเลอร์ AVR

3. ดับเบิลคลิกที่ตัวอุปกรณ์ไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ ATMEGA32 เพื่อให้โปรแกรมแสดงหน้าต่างการกำหนดคุณสมบัติของไอซีไมโครคอนโทรลเลอร์เบอร์ ATMEGA32 ตามรูปที่ 1.71

	ใบเนื้อหา		หน้าที่ 44
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004		หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์		

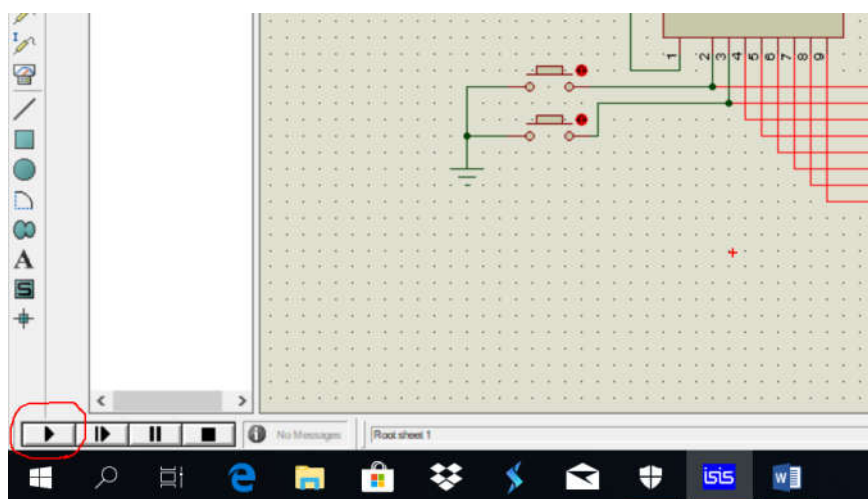
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์




รูปที่ 1.71 หน้าต่างการกำหนดคุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ ATMEGA32


4. จากรูปที่ 1.71 ในช่อง Program File ให้คลิกปุ่มไอคอนโฟลเดอร์เพื่อหาไฟล์โปรแกรมนามสกุล .hex ที่เราต้องการนำมา Simulate ของไมโครคอนโทรลเลอร์ ATMEGA32 ที่ได้จากการ Build ของโปรแกรม Atmel Studio 6.2 ส่วนในช่อง CKSEL Fuses ให้เลือกเป็น (1111) Ext. Crystal High Freq. และในช่อง Advanced Properties ในส่วนของ Clock Frequency ให้พิมพ์ค่า 16MHz หลังจากนั้นให้คลิกที่ปุ่ม OK

5. ทำการ Simulate วงจรโดยการคลิกที่ไอคอนตามรูปที่ 1.72



รูปที่ 1.72 แสดงตำแหน่งการคลิกปุ่ม play เพื่อเริ่มต้นการจำลองการทำงานของวงจรไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ ATMEGA32

	แบบฝึกหัด	หน้าที่ 1
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์		
<p><u>คำสั่ง</u> จงตอบคำถามต่อไปนี้ให้ถูกต้อง</p> <ol style="list-style-type: none"> จงอธิบายความหมายขอไมโครคอนโทรลเลอร์ จงเขียนโครงสร้างภายในของไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ที่ใช้สถาปัตยกรรมแบบ CISC และ RISC แตกต่างกันอย่างใด ประเภทของไมโครคอนโทรลเลอร์ถ้าแบ่งตามลักษณะการนำเข้าข้อมูลมาประมวลผล สามารถแบ่งได้กี่ประเภท อะไรบ้าง ขาพอร์ตของไมโครคอนโทรลเลอร์มีความสัมพันธ์กับประเภทของไมโครคอนโทรลเลอร์อย่างไร รีจิสเตอร์ TRISx ของไมโครคอนโทรลเลอร์ตระกูล PIC16F มีความสำคัญอย่างไร รีจิสเตอร์ DDRx ของไมโครคอนโทรลเลอร์ตระกูล AVR มีความสำคัญอย่างไร 		

	แบบฝึกหัด	หน้าที่ 2
	ชื่อวิชา ดิจิทัลและไมโครคอนโทรลเลอร์ รหัสวิชา 30127-2004	หน่วยที่ 1
	ชื่อหน่วย พื้นฐานของดิจิทัลและไมโครคอนโทรลเลอร์	
ชื่อเรื่อง พื้นฐานไมโครคอนโทรลเลอร์		
<p>8. ขาสัญญาณพอร์ตของไมโครคอนโทรลเลอร์เบอร์ AT89C51ED2 ,PIC16F887 และ ATMEGA32 มีการใช้งานแตกต่างกันอย่างไร</p> <p>.....</p> <p>.....</p> <p>.....</p> <p>9. ภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบ่งออกเป็นกี่กลุ่มอะไรบ้าง</p> <p>.....</p> <p>.....</p> <p>.....</p> <p>10. ภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล PIC16F แบ่งออกเป็นกี่กลุ่มอะไรบ้าง</p> <p>.....</p> <p>.....</p> <p>.....</p> <p>11. ภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล AVR แบ่งออกเป็นกี่กลุ่มอะไรบ้าง</p> <p>.....</p> <p>.....</p> <p>.....</p> <p>12. จงอธิบายองค์ประกอบของการเขียนโปรแกรมภาษาแอสเซมบลี</p> <p>.....</p> <p>.....</p> <p>.....</p>		