แผนการสอนประจำสัปดาห์ที่ 2

หัวข้อเรื่อง ไมโครโพรเซสเซอร์และไมโครคอนโทรลเลอร์
(Microprocessor and Microcontroller)

รายละเอียด

คอมพิวเตอร์มีการพัฒนาอย่างต่อเนื่องเริ่มตั้งแต่ยุคแรกที่มีการใช้หลอดสุญญากาศแทนวงจรในการคำนวณต่อมาในปี ค.ศ. 1954 ได้มีการคิดค้นการใช้ทรานซิสเตอร์แทนหลอดสุญญากาศ โดยบริษัท Taxas Instrument และในช่วงทศวรรษที่ 1950 นี้เองก็ได้มีการคิดค้นวงจรรวม (Integrated Circuit) ขึ้นมาใช้ในกับเครื่องคอมพิวเตอร์ จึงทำให้เครื่องคอมพิวเตอร์มีขนาดเล็กลงและใช้พลังงานไฟฟ้าน้อยลง มีความเร็วเพิ่มมากขึ้น และมีการพัฒนาวงจรรวมขนาดใหญ่ที่เรียกว่า Very Large Scale Integrated Circuit ทำให้สามารถรวมการประมวลผลทั้งหมดไว้ในชิพตัวเดียว

จำนวนชั่วโมงที่สอน 3 ชั่วโมง/สัปดาห์

กิจกรรมการเรียนการสอน

1. บรรยาย
2. สืบเสาะหาความรู้
3. ค้นคว้าเพิ่มเติม
4. ตอบคำถาม

สื่อการสอน

1. สื่ออิเล็กทรอนิกส์
2. เพาเวอร์พอยต์ พรีเซนเตชัน
3. บทเรียนออนไลน์
4. เอกสารอ้างอิงประกอบการค้นคว้า

แผนการประเมินผลการเรียนรู้

1. ผลการเรียนรู้

1.1 สังเกตจากงานที่กำหนดให้ไปทำมาส่ง

1.2 สังเกตจากการตอบคำถาม

1.3 สังเกตจากการนำความรู้ไปใช้

2. วิธีการประเมินผลการเรียนรู้

2.1 ตรวจผลงานภาคปฏิบัติ

2.2 ตรวจรายงาน

2.3 ตรวจแบบฝึกหัด

3. สัดส่วนของการประเมิน

3.1 ใบงานที่นักศึกษาทำมาส่ง

3.2 คะแนนเก็บในชั้นเรียน

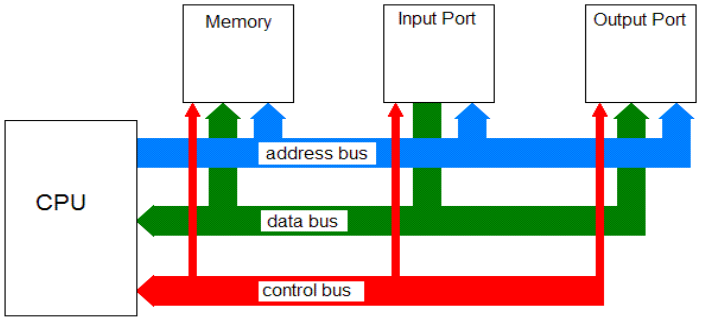
3.3 การเข้าชั้นเรียน

เนื้อหาที่สอน

ในสัปดาห์ที่ 2 การจัดการเรียนการสอน จะเกี่ยวข้องกับความแตกต่างระหว่างไมโครโพรเซสเซอร์และไมโครคอนโทรลเลอร์ ชนิดของไมโครโพรเซสเซอร์ โครงสร้างสถาปัตยกรรมของไมโครโพรเซสเซอร์ การเกิดขัดจังหวะ การระบุตำแหน่งในรีจีสเตอร์และการเข้าถึงข้อมูลในหน่วยความจำ ในขณะที่ คอมพิวเตอร์มีการพัฒนาอย่างต่อเนื่องเริ่มตั้งแต่ยุคแรกที่มีการใช้หลอดสุญญากาศแทนวงจรในการคำนวณ ซึ่งในเวลาต่อมาในปี ค.ศ. 1954 ได้มีการคิดค้นการใช้ทรานซิสเตอร์แทนหลอดสุญญากาศ โดยบริษัท Taxas Instrument และในช่วงทศวรรษที่ 1950 นี้เองก็ได้มีการคิดค้นวงจรรวม (Integrated Circuit) ขึ้นมาใช้ในกับเครื่องคอมพิวเตอร์ จึงทำให้เครื่องคอมพิวเตอร์มีขนาดเล็กลงและใช้พลังงานไฟฟ้าน้อยลง มีความเร็วเพิ่มมากขึ้น และมีการพัฒนาวงจรรวมขนาดใหญ่ที่เรียกว่า Very Large Scale Integrated Circuit ทำให้สามารถรวมการประมวลผลทั้งหมดไว้ในชิพตัวเดียว เช่น ไมโครโพรเซสเซอร์ตระกูลเพนเทียม (Pentium) ของบริษัทอินเทล (Intel Corporation) และสามารถรวมไมโครโพรเซสเซอร์หลายๆ ตัวให้สามารถทำงานรวมกันได้ นอกจากนั้นจำนวนข้อมูลที่ใช้ในการประมวลผลมีการพัฒนาขึ้นอย่างต่อเนื่องจาก 8 บิต มาเป็น 32 บิต 64 บิตและ 128 บิตตามลำดับ

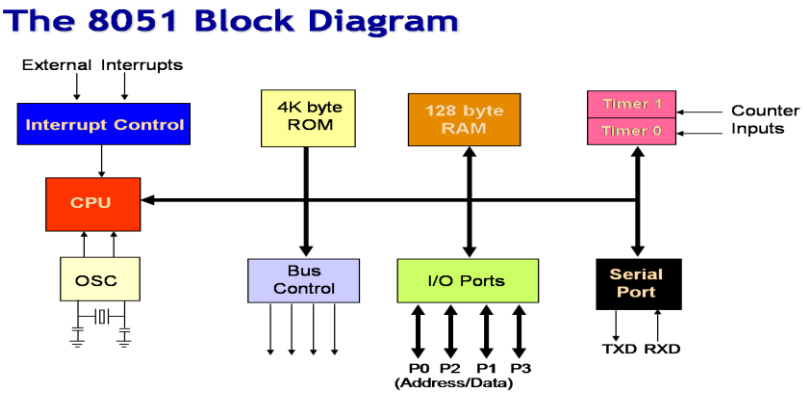
2.1 ไมโครโพรเซสเซอร์และไมโครคอนโทรลเลอร์

ไมโครโพรเซสเซอร์ (Microprocessor) เป็นอุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่ง ซึ่งมีลักษณะวงจรรวม (Integrated Circuit) หรือชิพ (Chip) โครงสร้างภายในจะมีวงจรรวมขนาดใหญ่ประกอบไปด้วย หน่วยคำนวณทางคณิตศาสตร์ (Arithmetic Unit) รีจิสเตอร์ (Register) บัสข้อมูล (Data Bus) บัสควบคุม (Control Bus) บัสที่อยู่ (Address Bus) รวมกันเป็นหน่วยประมวลผลกลาง (Central Processing Unit) เพื่อทำหน้าที่ในการประมวลผลตามโปรแกรมคำสั่งที่ป้อนเข้ามา แสดงได้ดังภาพที่ 2.1



ภาพที่ 2.1 แสดงบล็อกไดอะแกรมของไมโครโพรเซสเซอร์ (Block Diagram of Microprocessor)
ที่มา: (http://brainly.in/question/3843)

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชิพประมวลผลประเภทหนึ่งซึ่งทำหน้าตามโปรแกรมหรือชุดคำสั่งที่เขียนขึ้นมา ภายในประกอบด้วยวงจรรวมขนาดใหญ่ (Very Large Scale Integrated Circuit) ประกอบไปด้วย หน่วยคำนวณทางคณิตศาสตร์ (Arithmetic Unit) รีจิสเตอร์ (Register) บัสข้อมูล (Data Bus) บัสควบคุม (Control Bus) บัสที่อยู่ (Address Bus) รวมกันเป็นหน่วยประมวลผลกลาง (Central Processing Unit) หน่วยความจำ (Memory) วงจรนับ (Counter Circuit) วงจรจับเวลา (Timing Circuit) หรือวงจรอื่นๆ รวมอยู่ในชิพไมโครคอนโทรลเลอร์ แสดงได้ดังภาพที่ 2.2



ภาพที่ 2.2 แสดงบล็อกไดอะแกรมของโครคอนโทรลเลอร์ (Block Diagram of Microcontroller)
ที่มา: (https://aninditadhikary.wordpress.com/tag/microcontroller/)

2.2 ชนิดของไมโครโพรเซสเซอร์ จัดแบ่งตามลักษณะการใช้งานได้ 3 ประเภท

2.2.1 Dedicated or Embedded Controller เป็นไมโครโพรเซสเซอร์ที่ใช้ในอุปกรณ์อิเล็กทรอนิกส์โดยเฉพาะที่ เช่น ใช้ในการควบคุมการทำงานของเครื่องซักผ้า เตาไมโครเวฟ เครื่องคิดเลข และอื่นๆ โดยมากมักจะเรียกไมโครโพรเซสเซอร์ ประเภทนี้ว่า “ไมโครคอนโทรลเลอร์ (Micro Controller)” เช่น TMS-1000 ของบริษัท Texas Instrument สำหรับบริษัท Intel ได้เริ่มพัฒนาจาก ชิปขนาด 4 bits และในปี 1976 บริษัท Intel ได้เปิดตัว 8048 ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาด 8 bits ที่มี RAM, ROM และ I/O port ภายในชิปตัวเดียวกัน ในปัจจุบันมี 8096 ซึ่งมีทั้ง 16-bit CPU, RAM, ROM, UART, Port, Timer และ analog-to-digital converter ขนาด 16-bit ภายในชิปตัวเดียว

2.2.2 Bit-slice Processor เป็นไมโครโพรเซสเซอร์ที่ได้รับการพัฒนให้สามารถนำฟังก์ชันบางอย่างที่ไม่มีในไมโครคอนโทรลเลอร์ เช่น ฟังก์ชันการทำงานแบบ Multiplexer และ Sequencer เข้ามาทำงานบนชิพประมวลผลได้ อีกทั้งยังสามารถควบคุมการทำงานด้วยโปรแกรมชุดคำสั่ง โดยเรียกโปรแกรมชุดคำสั่งสำหรับฟังก์ชันการทำงานนี้ว่า “Micro Code”

2.2.3 General-purpose Processor เป็นไมโครโพรเซสเซอร์ที่ใช้ในอุปกรณ์อิเล็กทรอนิกส์โดยทั่วไป ซึ่งควบคุมการทำงานด้วยโปรแกรม แบ่งตามรุ่นที่พัฒนาโดยบริษัทอินเทล ((Intel Corporation) ผู้ผลิต โดยมีการพัฒนามาตั้งแต่รุ่น 4040 ซึ่งเป็นไมโครโพรเซสเซอร์ขนาด 4 บิต และพัฒนาเป็นรุ่น Pentium จำแนกได้ดังนี้

- **8086** มี data bus ขนาด 16 bits ดังนั้นจึงสามารถ อ่าน/เขียนข้อมูลได้ครั้งละ 16 bit 8086 มี data bus ขนาด 16 bits ดังนั้นจึงสามารถ อ่าน/เขียนข้อมูลได้ครั้งละ 16 bits หรือ 8 bits

- **8086** มี address bus ขนาด 20 bits ดังนั้นจึงสามารถระบุตำแหน่งในหน่วยความจำได้สูงสุด 220 หรือ 1,048,576 ตำแหน่ง โดยแอดเดรสแต่ละตำแหน่งจะจัดเก็บข้อมูลขนาด 8 bits ข้อมูลขนาด 16 bit word จะถูกจัดเก็บไว้ในหน่วยความจำ ตำแหน่งที่แอดเดรสอยู่ต่อจากกัน ดังนั้น ถ้าไบต์แรกของ word ถูกจัดเก็บไว้ใน memory address เลขคู่ 8086 จะสามารถอ่านได้ภายในการปฏิบัติเพียงครั้งเดียว แต่ถ้าไบต์แรกของ word ถูกจัดเก็บไว้ใน memory address เลขคี่ 8086 จะอ่านข้อมูลนั้นโดยใช้ Bus Operation 2 ครั้ง (อ่านทีละ byte)
- **8088** เป็นไมโครโพรเซสเซอร์ขนาด 16 bit (ALU, รีจิสเตอร์ภายใน และชุดคำสั่ง (Instruction) จะมีขนาด 16
  bit word เช่นเดียวกับ 8086)

– 8088 มี data bus ขนาด 8 bits ดังนั้นจึงสามารถอ่าน หรือเขียนข้อมูลได้ครั้งละ 8 bits เท่านั้น ในการอ่าน/เขียนข้อมูลหรือชุดคำสั่งขนาด 16 bits จะต้องใช้ Bus Operation 2 ครั้ง

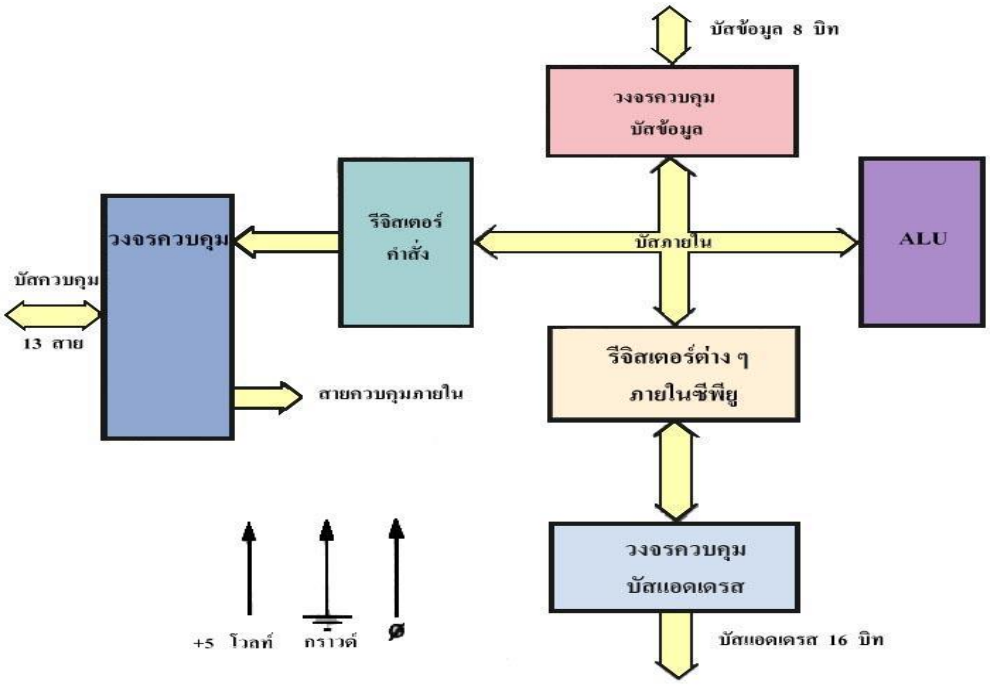
– 8088 มี address bus ขนาด 20 bits ดังนั้นจึงสามารถ address ตำแหน่งในหน่วยความจำได้สูงสุด 220 หรือ 1,048,576 ตำแหน่ง

- **80186 และ 80188** เป็นไมโครโพรเซสเซอร์ที่ปรับปรุงมาจาก 8086 และ 8088 นอกจากทั้งคู่จะเป็น CPU ขนาด 16 bit แล้ว ยังมี programmable peripheral devices บูรณาการรวมอยู่ใน package เดียวกัน ชุดคำสั่งต่างๆ ใน 80186 และ 80188 เป็น superset ของชุดคำสั่งของ 8086 และ 8088 ซึ่งหมายความว่าชุดคำสั่งเดิมใน 8086 และ 8088 ยังคงสามารถทำงานได้ใน 80186 และ 80188 โดยมีชุดคำสั่งเพิ่มเติมไม่มากนัก ดังนั้นโปรแกรมที่เขียนให้สามารถทำงานได้ใน 8086 หรือ 8088 จะยังคงสามารถทำงานได้ใน 80186 และ 80188 (เรียกว่า upward compatible) แต่โปรแกรมที่เขียนให้ทำงานบน 80186 และ 80188 อาจจะไม่สามารถทำงานได้ใน 8086 และ 8088
- **80286** เป็นไมโครโพรเซสเซอร์ ที่ปรับปรุงมาจาก 8086 (มีการปรับปรุงในระดับสูง) ที่ได้รับการ ออกแบบให้ใช้เป็น CPU ในเครื่องไมโครคอมพิวเตอร์ที่มีความสามารถในการทำงานแบบ Multi–user/ Multitasking การทำงานของ 80286 ใน real address mode จะมีฟังก์ชันการทำงานเหมือน 8086 ความเร็วสูงการทำงานใน virtual address mode จะมีคุณสมบัติในการแยกโปรแกรมประยุกต์ของผู้ใช้ออกจากระบบปฏิบัติการ เพื่อไม่ให้ส่วนของโปรแกรมไปทำลายโปรแกรมระบบได้

- **80386 และ 80486** เป็นไมโครโพรเซสเซอร์ ขนาด 32 bit ซึ่งสามารถระบุตำแหน่งของแอดเดรสในหน่วยความจำได้สูงสุดถึง 4GB และมีการนำเทคโนโลยี RISC (Reduced Instruction Set Computer) มาใช้ ทำให้มีคุณสมบัติที่ซับซ้อนเพิ่มมากขึ้นในเรื่องของการทำงานแบบ Multi–user/ Multitasking

2.3 โครงสร้างสถาปัตยกรรมของไมโครโพรเซสเซอร์

ไมโครโพรเซสเซอร์ตระกูลของ Intel จะมีโครงสร้างสถาปัตยกรรมในลักษณะเดียวกัน สำหรับการเริ่มต้นศึกษาโครงสร้างสถาปัตยกรรมของไมโครโพรเซสเซอร์จาก 8086 เพื่อให้เข้าใจได้โดยง่าย และไม่ซับซ้อนมากนักและก่อนที่จะเข้าไปถึงการเขียนโปรแกรมให้สามารถทำงานบนวิพตระกูล 8086 จะต้องทราบโครงสร้างการทำงานภายใน ไม่ว่าจะเป็น ALU, Flag, Register, Instruction Byte Queue และ Segment Register 38 CPU 8086 มีแอดเดรส 20 เส้น (Address Bus 20 bits) ทำให้สามารถอ้างถึงแอดเดรสได้ 1,045,576 ไบต์ (หรือ 1 MB – เมกะไบต์) หรือมากเป็น 16 เท่า ของไมโครโปรเซสเซอร์ขนาด 8 บิต และมี data bus ขนาด 16 บิต จึงทำให้ CPU 8086 สามารถเรียกข้อมูลได้ทีละ 1 เวิร์ด หรือ 2 ไบต์ และยังเตรียมพร้อมสำหรับการทำงานแบบหลาย CPU รวมทั้งการทำงานร่วมกับอุปกรณ์อื่นได้ด้วย แสดงได้ดังภาพที่ 2.3



ภาพที่ 2.3 แสดง Block Diagram ของ ซีพียู (CPU 8086)

ที่มา: (https://sites.google.com/site/kwantharathepsiri67/3)

8086 แบ่งส่วนตามฟังก์ชันการทำงานที่เป็นอิสระต่อกันออกเป็น 2 ส่วน คือ BIU (Bus Interface Unit) และ EU (Execution Unit) เพราะจะทำให้ความเร็วในการทำงานสูงขึ้น

1. BIU จะทำหน้าที่ส่งค่าแอดเดรสเพื่อไป fetch คำสั่งจากหน่วยความจำ, อ่านข้อมูลจากหน่วยความจำ และพอร์ต, และเขียนข้อมูลไปยังหน่วยความจำ และพอร์ต

2. EU ของ 8086 จะบอก location ที่จะไป fetch instruction, decode instruction และ execute instruction

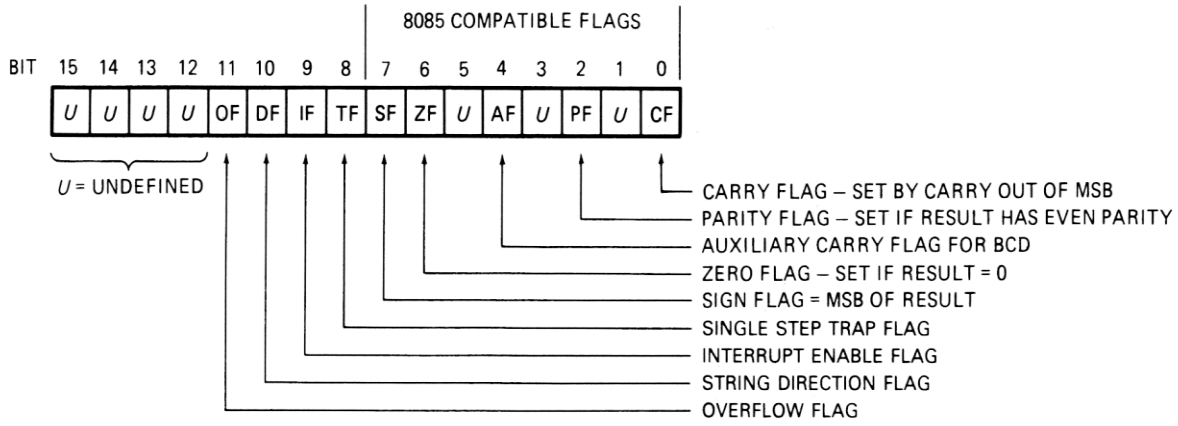
2.3.1 EU (Execution Unit) ประกอบด้วย

1. Control Circuitry ทำหน้าที่ควบคุม Operation ภายในทั้งหมด

2. Instruction Decoder ทำหน้าที่แปลความหมายของชุดคำสั่งที่นำมาจาก Memory ให้เป็นลำดับของภาษาเครื่องที่ EU สามารถปฏิบัติได้

3. ALU มีขนาด 16 bits ซึ่งมีฟังก์ชันในการ Add (บวก), Subtraction (ลบ), AND, OR, XOR, increment, decrement, complement และ shift

4. Flag Register เป็น flip–flop ที่แสดงสถานะของการ execute instruction หรือการควบคุมการทำงานของชุดคำสั่ง Flag Register ขนาด 16 บิต โดยที่ใน EU จะบรรจุ active flag 9 ตัว โดยเป็น flag แสดงสถานะ (Condition flags) จำนวน 6 บิต และ flag ควบคุม (Control flags) จำนวน 3 บิต สำหรับบิตส่วนที่เหลือก็ไม่ได้ใช้ แสดงได้ดังภาพที่ 2.4



ภาพที่ 2.4 แสดงสถานะแฟล็ก (Flag Status) ที่ใช้ในการตรวจสอบการทำงานของซีพียู (CPU 8086)
ที่มา: (http://www.cpe.ku.ac.th/~yuen/204323/gen_arch/86register.html)

4.1 แฟล็กแสดงสถานะ (Conditional Flag) มีทั้งหมด 6 ตัว จะถูกใช้ในการแสดงเงื่อนไขบางอย่างที่สร้างโดยชุดคำสั่งในการ SET หรือ RESET ค่า Flag เหล่านี้ จะดำเนินการโดย EU บนพื้นฐานของผลลัพธ์ของการคำนวณ เช่น carry out flag จะถูก set ให้มีค่าเป็น 1 หากผลลัพธ์ของการบวกเลขฐาน 2 จำนวน 16 bits มีตัวทดหลักเพิ่มเติมจากบิตที่มีความสำคัญสูงสุด หรือ MSB (Most Significant Bit)

1. CF (Carry Flag) จะถูก SET เมื่อมี CARRY OUT ของ MSB
2. PF (Parity Flag) จะถูก SET เมื่อผลลัพธ์มี parity เป็น EVEN
3. AF (Auxiliary Carry Flag) สำหรับ BCD
4. ZF (Zero Flag) จะถูก SET เมื่อผลลัพธ์ = 0
5. SF (Sign Flag) เป็นค่า MSB ของผลลัพธ์ (0 เป็นค่า +, 1 เป็นค่าลบ)
6. OF (Overflow Flag) จะถูก SET เมื่อผลลัพธ์เกิดการ Overflow

4.2 แฟล็กควบคุม (Control Flag) มีทั้งหมด 3 ตัว จะถูกใช้ในการควบคุมการปฏิบัติที่แน่นอนของโพรเซสเซอร์ ซึ่ง
แฟล็กเหล่านี้จะถูก SET หรือ RESET โดยเจตนาของชุดคำสั่งที่ใส่ไว้ในโปรแกรม

1. TF (Trap Flag) ถูกนำมาใช้สำหรับชุดคำสั่งทำงานในลักษณะ Single Step
2. IF (Interrupt Flag) ถูกนำมาใช้กำหนดการขัดจังหวะในชุดคำสั่งของโปรแกรม
3. DF (Direction Flag) ถูกนำมาใช้กับชุดคำสั่งที่เป็นสายอักขระ (String)

รูปแบบของแฟล็ก (flag) ได้ถูกออกแบบมาเพื่อแสดงผลของการกระทำทางคณิตศาสตร์และตรรกะ สถานะที่เป็นไปได้ของ flag จะมีได้แค่สองสถานะ คือ ถูกเซตทำให้มีค่าเป็น 1 หรือถูก รีเซตให้มีค่าเป็น 0 แฟล็กตัวทด (carry flag) จะสดงลักษณะบอกการทดหรือ การขอยืมของบิตสูงสุดของผลลัพธ์ , แฟล็กพาริตี้ (parity flag) จะใช้แสดงจำนวนคู่หรือคี่ของตัวเลขที่เป็น 1 ในรีจิสเตอร์ ถ้าเป็นคู่ flag นี้จะถูกเซตเป็น 1, แฟล็กตัวทดช่วย (auxiliary carry flag) จะใช้งานแบบ "decimal adjust" ในการคำนวณ BCD, แฟล็กศูนย์ (zero flag) จะมีค่าเป็น 1 เมื่อผลลัพธ์เป็นศูนย์แฟล็กเครื่องหมาย (sign flag) จะได้รับการเซตเมื่อผลลัพธ์เป็นเลขจำนวนลบ , โอเวอร์โฟลว์แฟล็ก (OF) ใช้เพื่อแสดงว่ามีความผิดพลาดในการคำนวณ เนื่องจากการใส่เครื่องหมายของผลลัพธ์ เช่นถ้ามีการบวกเลข +127 กับ+2 ผลลัพธ์ที่ได้ถ้าคิดคำนวณแบบมีเครื่องหมายจะได้คำตอบเป็น –127 ซึ่งผิดความหมาย เมื่อเกิดเหตุการณ์ในลักษณะนี้ 8086 จะเซตโอเวอร์โฟลว์แฟล็กให้เป็น 1

แฟล็กควบคุมทิศทาง (direction flag) จะกำหนดทิศทางของกลุ่มคำสั่งเกี่ยวกับสตริง ว่าจะมีทิศทางการทำงานจากแอดเดรสมากไปน้อยหรือจากแอดเดรสน้อยไปมาก

แฟล็กขัดจังหวะ (IE) ถ้าถูกเซตเป็น 1 แสดงว่าจะยอมรับขัดจังหวะจากภายนอก โดยหยุดการทำงานตามปกติของโปรแกรมไว้ก่อน และ Tap flag (TF) จะทำให้ 8086 ทำงานแบบทีละคำสั่งเพื่อการแก้ไขโปรแกรม

2.3.2 BIU (Bus Interface Unit) ประกอบด้วย

1. Queue : BIU จะจัดเก็บชุดคำสั่งที่ fetch มาจากหน่วยความจำได้สูงสุด 6 คำสั่ง โดยจะใช้เทคโนโลยีที่เรียกว่า FIFO (First–In First–Out) เมื่อ EU พร้อมที่จะทำงานกับชุดคำสั่งต่อไปก็เพียงแต่นำข้อมูลและชุดคำสั่งมาจาก Queue โดยไม่ต้องไปยุ่งเกี่ยวกับ Bus จึงสามารถทำงานได้เร็วขึ้นอีกหลายเท่า แทนที่จะต้องรอการอ่านข้อมูลและชุดคำสั่งมาจากหน่วยความจำภายนอก เว้นแต่ว่าเมื่อเมื่อพบคำสั่ง JMP (Jump) หรือ CALL ที่ต้องเรียกข้อมูลมาจากแอดเดรสที่กำหนดไว้ในโปรแกรม คุณสมบัติในการ Fetch คำสั่งต่อไปมาเก็บไว้ใน Queue ในขณะที่ EU กำลังexecute คำสั่งอื่นอยู่ เรียกว่า Pipelining

2. Segment Register : BIU จะส่งค่า address ขนาด 20 bits ออกไป ดังนั้นจึงสามารถที่จะระบุตำแหน่งใน Memory ได้ 220 = 1,045,576 bytes แต่อย่างไรก็ตาม 8086 จะสามารถทำงานกับ segment 4 ส่วน ในหน่วยความจำ (ส่วนละ 65,536 bytes = 64 Kbyte) ดังนั้นภายในหน่วยความจำขนาด 1 Mbyte (1,045,576 bytes) จึงต้องมี segment register 4 ตัว เพื่อจัดเก็บค่าแอดเดรสจุดเริ่มต้นของแต่ละ Segment ในหน่วยความจำ ที่ 8086 ใช้ในการทำงาน

1. CS (Code Segment)
2. SS (Stack Segment)
3. ES (Extra Segment)
4. DS (Data Segment)

Segment Register ถูกใช้ในการจัดเก็บค่าแอดเดรสขนาด 16 bits ของจุดเริ่มต้นของแต่ละส่วน ตัวอย่างเช่น Code Segment Register จะจัดเก็บค่าแอดเดรสเริ่มต้นของหน่วยความจำส่วนที่ BIU fetch instruction มา โดย BIU จะใส่ค่า 0 ให้กับ 4 bits สุดท้ายของแอดเดรสขนาด 20 bits เช่นถ้า code segment register มีค่า 348A (Hex) แสดงว่าส่วนในหน่วยความจำ ที่จัดเก็บโปรแกรมจะเริ่มต้นที่ address 348A0 (Hex) จึงหมายความว่าแต่ละส่วนขนาด 64 K สามารถที่จะอยู่ตำแหน่งใดในหน่วยความจำขนาด 1 MB โดยที่ Stack เป็นส่วนของหน่วยความจำที่แยกไว้ต่างหาก เพื่อจัดเก็บค่าแอดเดรสเริ่มต้นของโปรแกรมย่อยและเก็บข้อมูลที่ใช้ในโปรแกรมย่อย ในขณะที่ทำการ execute Subprogram หรือ Procedure

2.4 การเกิดการขัดจังหวะ (Interrupt Handle)

การขัดจังหวะของ 8086 อาจจะมาจากหลาย ๆ แหล่งได้ เช่น การขัดจังหวะที่มาจากอุปกรณ์ภายนอกการขัดจังหวะทาง software หรือการขัดจังหวะที่มาจากตัว CPU 8086 เองก็ได้ การขัดจังหวะจากอุปกรณ์ภายนอกจะเข้ามาทางขา 2 ขา คือ INTR (interrupt request) และ NMI(non-maskable interrupt) โดยที่ INTR จะถูกกำหนดให้ทำงานได้โดยคำสั่ง IE (interrupt enable flag) ที่เป็นส่วนหนึ่งของ 8086 flag ถ้า flag IE ถูกเซตจะ ทำให้ 8086 สามารถจะรับการขัดจังหวะจาก INTR ได้ แต่ถ้า IF ถูกเคลียร์ 8086 จะไม่รับการขัดจังหวะจาก INTR ส่วน NMI จะเป็นการขัดจังหวะที่ไม่สามารถจะหยุดได้ มักจะใช้ในกรณีที่เกิดเหตุการณ์สำคัญ เช่น เกิดความผิดพลาดของพาริตี้หน่วยความจำ (memory parity) หรือเกิดไฟตก เป็นต้น

INTO (interrupt overflow) คือ การขัดจังหวะของ 8086 ที่เกิดจากข้อผิดพลาดของการหารและการทำงานทีละขั้น ข้อผิดพลาดจากการหาร (divide error) เกิดขึ้นเนื่องจากผลของการหารอาจจะมีค่ามากกว่าเนื้อที่ซึ่งเตรียมไว้เป็นที่เก็บผลลัพธ์ หรืออาจจะเกิดจากการหารด้วยเลข 0 ส่วนลักษณะของการทำงานทีละขั้น (single step) จะเป็นการขัดจังหวะที่เกิดเนื่องจากการ execute คำสั่งทีละคำสั่ง เหตุการณ์นี้เกิดขึ้นเนื่องจาก TF (trap flag) ในรีจิสเตอร์แฟล็กถูกเซตให้เป็น 1

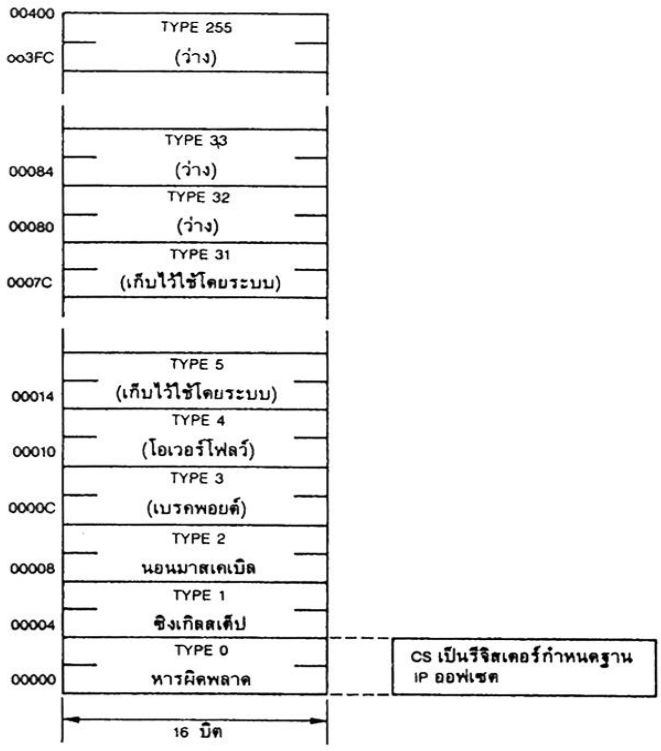
เมื่อเกิดการขัดจังหวะเหล่านี้ขึ้นมาแล้วก็จะเกิดเหตุการณ์เฉพาะอย่างขึ้นถ้ามีการขัดจังหวะจากภายนอก คำสั่งชุดสุดท้ายจะถูก execute จนเสร็จสิ้นก่อน (ยกเว้น ถ้าเป็นคำสั่ง MOV หรือ POP จะมีการ execute คำสั่งต่อไปก่อนแล้วจึงหยุด) ขัดจังหวะทุกตัวจะทำการเก็บค่าของแฟล็กรีจิสเตอร์ไว้ในสแตก เสร็จแล้ว 8086 จะทำการรีเซตแฟล็ก IF และ TF ให้เป็น 0 ทั้งนี้เพื่อป้องกัน INTR อินพุตที่เข้ามาใหม่ และป้องกันการเกิดการทำงานทีละขั้นในขณะที่อยู่ในโปรแกรมขัดจังหวะ เนื่องจากการที่แฟล็กถูกเก็บค่าไว้หลังจากโปรแกรมขัดจังหวะสิ้นสุดแล้วค่าต่าง ๆ ที่เก็บไว้จะถูกดึงกลับมาใช้งานได้ จากนั้น 8086 จะทำการเก็บค่าของ CS และ IP ลงบนสแตกเพื่อที่จะเก็บตำแหน่งของแอดเดรสส่งกลับ (return address) ในลักษณะเดียวกับคำสั่ง CALL และในขั้นตอนสุดท้าย CS และ IP จะถูกอ่านจากตาราง interrupt vector (ขึ้นกับหมายเลขการขัดจังหวะ) ตาราง interrupt vector ก็คือบริเวณของเนื้อที่ของหน่วยความจำเริ่มตั้งแต่แอดเดรส 0000 จำนวน หนึ่งกิโลไบต์ แสดงได้ดังภาพที่ 2.5

การขัดจังหวะแต่ละตัวจะชี้ไปยังตำแหน่งที่แน่นอนและสามารถที่จะกระโดดไปถึงที่กำหนดได้โดยอัตโนมัติ เมื่อเกิดการขัดจังหวะจากลักษณะของ divide error จะทำให้เกิดการขัดจังหวะ 0, single-step จะเป็นการขัดจังหวะ 1, การขัดจังหวะแบบ Non-maskable เป็นแบบการขัดจังหวะ 2 , bread-point จะเกิดการขัดจังหวะ 3 , และการเกิด over flow จะเป็นการขัดจังหวะ 4

ส่วน ขัดจังหวะ 5 ถึง 31 เตรียมไว้สำหรับ ผู้ใช้จะกำหนดเอาเองภายหลัง ส่วนการขัดจังหวะที่เหลือเราอาจจะควบคุมโดยใช้คำสั่ง INTR หรือ INT ได้

สำหรับ INTR จะตอบรับสัญญาณการขัดจังหวะจากภายนอก ส่วน INT จะรับได้โดยตัวคำสั่ง นั่นเอง CPU 8086 สามารถที่จะรับรู้ชนิดของการขัดจังหวะได้โดยการคูณค่าของคำสั่ง ด้วย 4 ผลลัพธ์ที่ได้จะบอกตำแหน่งของตารางขัดจังหวะ ซึ่งค่าของตารางจะถูกส่งค่าไปยัง CS และ IP ค่าต่าง ๆในตาราง เราจะต้องเป็นผู้กำหนดโดยกำหนดเป็นค่าของเซกเมนต์และค่าของออฟเซตที่แสดงตำแหน่งของการขัดจังหวะรูทีน เช่น การขัดจังหวะที่แอดเดรส 0–3 จะถูกต้องกำหนดเป็นค่าบอกแอดเดรสของ CS และ IP เพื่อจะชี้รูทีนของ divide error เป็นต้น

การขัดจังหวะรูทีน จะต้องเริ่มต้นด้วยการเก็บค่าของรีจิสเตอร์ต่าง ๆไว้ก่อน เพราะเราไม่ทราบว่าจะเกิดขัดจังหวะเมื่อถึงจุดใด และเราไม่ต้องการทำให้ค่ารีจิสเตอร์ต่าง ๆ เสียไปและเราต้อง ปิดท้ายรูทีนด้วยคำสั่งIRET คำสั่งนี้จะทำงานเหมือนคำสั่ง RET ธรรมดาแต่จะทำการอ่านค่าของแฟล็กออกมาจากสแตกด้วย (ศัพท์บัญญัติ ราชบัณฑิตยสถาน, 2544)



ภาพที่ 2.5 แสดงการเกิดขัดจังหวะ ที่ตำแหน่ง 0000– 03FF ขนาด 1 กิโลไบต์ (1Kbytes)
ที่มา: (http://www.cpe.ku.ac.th/~yuen/204323/gen_arch/86interrupt.html)

2.5 รีจีสเตอร์ของซีพียู 8086 (CPU 8086 Register)

CPU 8086 ประกอบด้วยรีจิสเตอร์ที่เกี่ยวข้อง แบ่งออกเป็น 2 กลุ่ม (แสดงได้ดังภาพที่ 2.6) ดังนี้

1. กลุ่มข้อมูล (Data Group Register) แบ่งออกเป็น

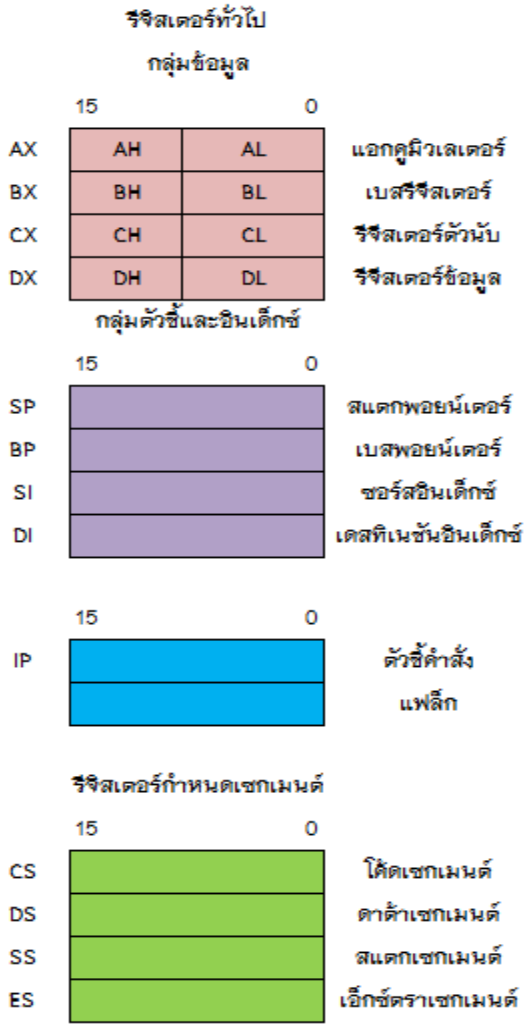
1.1 รีจีสเตอร์ทั่วไป (General–Purpose Register)

1.2 รีจิสเตอร์ตัวชี้ (Instruction Pointer Register)

1.3 แฟล็ก (flag)

2. กลุ่มกำหนดเซกเมนต์ (Segment Register) แบ่งออกเป็น

2.1 รีจิสเตอร์กำหนดเซกเมนต์ (Segment Register)



ภาพที่ 2.6 แสดงรีจีสเตอร์ที่ใช้ในไมโครโพรเซสเซอร์ 8086

ที่มา: (http://www.shsu.edu/~csc_tjm/spring2005/cs272/8086.html)

1.1 รีจีสเตอร์ทั่วไป (General–Purpose Register) รีจิสเตอร์ที่มีอยู่ทั้งหมดของ 8086 เป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งกลุ่มข้อมูลจะแบ่งย่อยออกเป็น

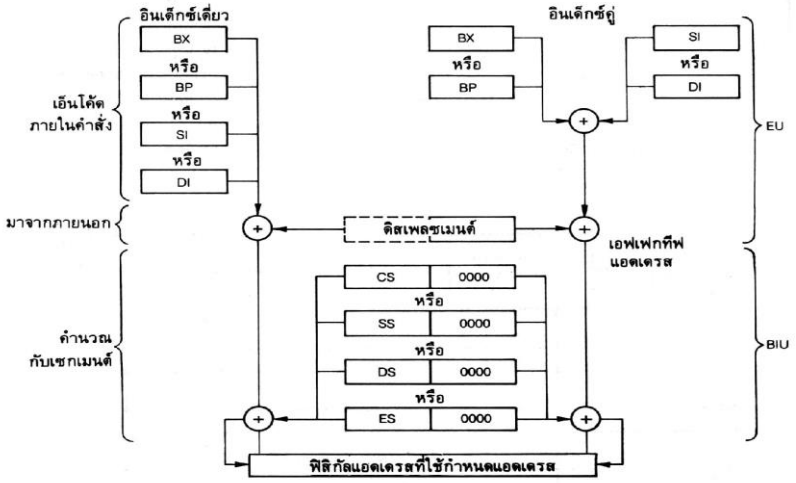
1.1.1 รีจิสเตอร์ AX (Accumulator) หรือแอกคูมิวเลเตอร์ ซึ่งจะประกอบด้วย AH และ AL โดย AX ทำหน้าที่ในการใช้คูณหารหรือเป็นรีจิสเตอร์เกี่ยวกับอินพุตเอาต์พุตที่เป็นเวิร์ดส่วน AL ทำหน้าที่ใช้ในการคำนวณ คูณหาร หรือทำงานเกี่ยวกับอินพุต (Input) เอาต์พุท (Output) แปลง ข้อมูลจัดการคำนวณแบบตัวเลขฐานสิบแบบ 8 บิต และ AH ใช้เป็นรีจิสเตอร์ในการคูณและหารได้

1.1.2 รีจิสเตอร์ BX (Base Register) หรือเบสรีจิสเตอร์ใช้ในการแปลงข้อมูล

1.1.3 รีจิสเตอร์ CX (Counter Register) หรือรีจิสเตอร์ตัวนับใช้ในคำสั่งจัดการเกี่ยวกับสตริง และการทำลูป CL ใช้เป็นตัวแปรสำหรับการเลื่อนบิตหรือหมุนบิต

1.1.4 รีจิสเตอร์ DX (Data Register) หรือรีจิสเตอร์ข้อมูลใช้ในคำสั่งคูณ หารเป็นเวิร์ด (Words) หรือใช้ในรูปแบบอ้างอินพุตเอาต์พุตแบบอ้อม แสดงได้ดังภาพที่ 2.6

1.2 รีจิสเตอร์ตัวชี้ (Instruction Pointer Register) รีจิสเตอร์ใช้งานทั่วไปในกลุ่มตัวชี้และอินเด็กซ์ประกอบด้วย SP หรือสแต็กพอยน์เตอร์ใช้กับคำสั่งสแตก สำหรับรีจิสเตอร์ต้นทางหรือ SI (Source Index) รีจิสเตอร์ ปลายทางหรือ DI (Destination Index) จะใช้กับคำสั่งที่เกี่ยวกับสตริง และเบสพอยน์เตอร์หรือ BP (base pointer) รีจิสเตอร์เหล่านี้สามารถที่จะนำมาใช้เป็นตัวอ้างถึง Physical address ได้ด้วยวิธีการต่างๆ ดังแสดงได้ดังภาพที่ 2.7

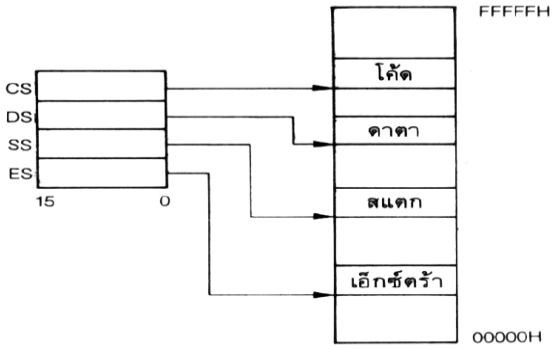


ภาพที่ 2.7 แสดงโครงสร้างรีจีสเตอร์ 8086 กับการคำนวณหาค่าตำแหน่งในหน่วยความจำ

ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

อธิบายจากภาพที่ 2.7 รีจิสเตอร์ขนาด 16 บิต 4 ตัว ในกลุ่มข้อมูล ได้แก่ AX (Accumulator), BX (Base), CX (Counter) และ DX (Data) ยังถูกแบ่งออกเป็นครึ่งบน (AH, BH, CH และ DH) และครึ่งล่าง (AL, BL, CL และ DL) รีจิสเตอร์เหล่านี้อาจจะทำงานในลักษณะ เป็นเวิร์ดหรือไบต์ก็ได้ ในกลุ่มนี้ถือว่าเป็นรีจิสเตอร์ ข้อมูลรีจิสเตอร์ในกลุ่ม pointer และ index มักจะนำไปใช้ในการอ้าง Physical address และยังใช้ในการปฏิบัติทางคณิตศาสตร์และลอจิกได้รีจิสเตอร์ BP และ SP ใช้ในการชี้ค่าตำแหน่งในสแตก (Stack Address) เพื่อจะทำการเก็บตำแหน่ง address กลับจากโปรแกรมย่อย (Subprogram) โดยใช้หลักการเหมือนสแตก (Stack) ของ CPU 8 บิต ส่วน BP นั้นจะใช้สำหรับเป็น ตัวบวก เพื่อชี้ค่าสแตกอีกทีหนึ่ง เพื่อจะเป็นวิธีที่จะอำนวยความสะดวกในการส่งค่า พารามิเตอร์เข้าสู่โปรแกรมโดยผ่านทางสแตก ซึ่งช่วยในการส่งค่าของตัวแปรระหว่างโปรแกรมได้ดี รีจิสเตอร์ SI และ DI จะทำการติดต่อกับหน่วยความจำได้ ซึ่งจะใช้งานได้ดีในกลุ่มคำสั่งพวก สตริงและการ ทำงานเป็น block หรือ การเชื่อมโยงข้อมูลในโครงสร้างแบบ array ในการ fetch คำสั่งของ 8086 จะใช้ IP (instruction pointer) ขนาด 16 บิต เป็นตัวชี้ตำแหน่ง address ของคำสั่งถัดไปที่จะถูกทำงาน ซึ่งความจริงแล้วค่าใน IP ไม่ได้กำหนดตำแหน่งโดยตรง แต่จะมีการคำนวณมาก่อน

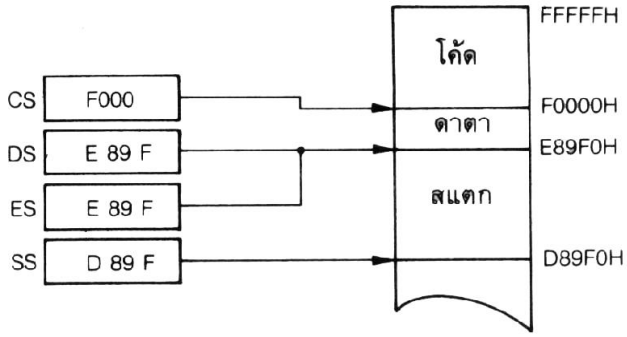
2. รีจิสเตอร์กำหนดเซกเมนต์ (Segment Register) ประกอบด้วยรีจีสเตอร์ย่อยมี 4 ตัว คือ CS (code segment), DS (data segment), SS (stack segment) และ ES (extra segment) ซึ่งจะนำมาใช้ในหลักการแบบบล็อกแอดเดรสของ 8086 หน่วยความจำ (Memory) จะได้รับการจัดสรรเป็นส่วนใหญ่ ๆ คือ code (หรือชุดคำสั่ง), ข้อมูล (ตัวเลข หรือตัวอักษร) และสแตกสำหรับการเก็บค่าแอดเดรสกลับคืนจากโปรแกรมย่อยเพื่อที่จะป้องกันการสับสน จึงกำหนดค่าให้แยกกันได้ซึ่ง 8086 จะมองลักษณะของหน่วยความจำ โดยแบ่งหน่วยความจำเป็นกลุ่ม ๆ ในรูปแบบของ เซกเมนต์ ในหนึ่งเซกเมนต์จะชี้ได้ถึง 64 กิโลไบต์ โดยเซกเมนต์รีจีสเตอร์ทั้งสี่ตัว จะแสดงแอดเดรสเริ่มต้นของหน่วยความจำที่จะติดต่อด้วย สังเกตได้จากภาพที่ 2.8 รีจีสเตอร์ CS จะบรรจุค่าแสดงแอดเดรสเริ่มต้นของโปรแกรม, DS จะเก็บค่า Data segment ในขณะนั้น,SS จะเก็บค่า Stack segment ในขณะนั้น และ ES จะกำหนด Segment ของข้อมูลรวมที่เรียกว่า Global Data Segment แสดงได้ดังภาพที่ 2.8



ภาพที่ 2.8 แสดงการแบ่งกลุ่มของ Segment Register

ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

เซกเมนต์จะแสดงตำแหน่งเหมือนกับ Paragraph โดยจะเลื่อนไปทางซ้าย 4 บิต เพื่อที่จะกำหนด หรืออ้างแอดเดรสให้ครบ 20 เส้น โดยจุดเริ่มต้นของ Paragraph จะต้องมี 4 บิต หลังสุดเป็น 0 เช่น เป็น 00000H, 00010H , 00020H เป็นต้น จากรูป ค่าของ SS จะชี้ตำแหน่งของเซกเมนต์ค่า D89F0 สังเกตเห็นว่าส่วนพื้นที่ ของหน่วยความจำอาจจะสลับหรือใช้บริเวณเดียวกันของหน่วยความจำได้ แสดงได้ดังภาพที่ 2.8



ภาพที่ 2.9 แสดงตำแหน่งเหมือนกับ Paragraph ใน Segment Register

ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

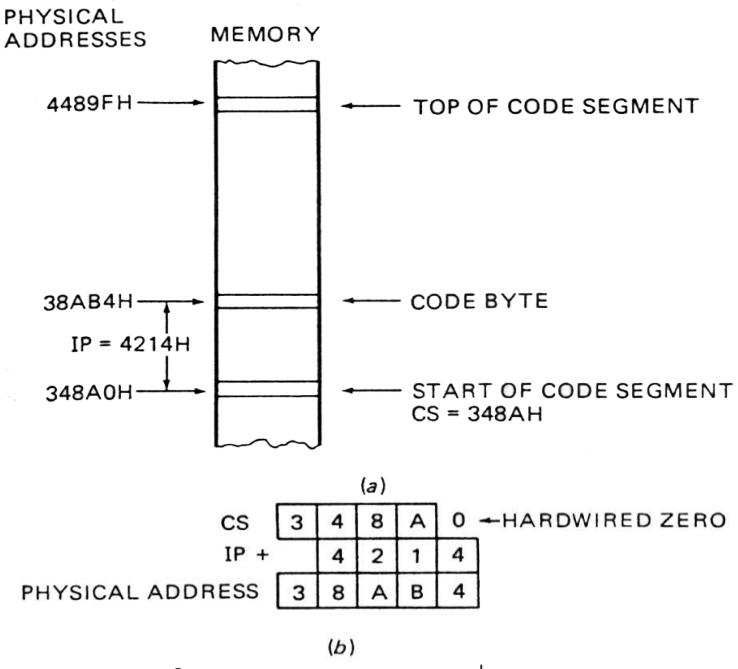
เพื่อที่จะทำการติดต่อกับข้อมูลหนึ่งไบต์หรือหนึ่งเวิร์ดนั้น 8086 ได้เตรียมค่าออฟเซต เพื่อใช้อ้างตำแหน่งตั้งแต่จุดเริ่มต้นของเซกเมนต์แอดเดรส ตำแหน่งใด ๆ จะได้มาจากการบวกค่าเซกเมนต์ รีจิสเตอร์กับค่าของออฟเซต 16 บิต เช่นถ้าเซกเมนต์มีค่า E89F จะให้ออฟเซตมีค่า 0003H จะทำให้การอ้างแอดเดรสไปที่ 89F3 การจะใช้เซกเมนตุรีจิสเตอร์และค่าออฟเซตตัวใดนั้นจะขึ้นอยู่กับชนิดของคำสั่งด้วย ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 แสดงชนิดของการอ้างอิงหน่วยความจำในเซกเมนต์รีจีสเตอร์

ชนิดของการอ้างอิงหน่วยความจำ	ค่าเซกเมนต์ปกติ	ค่าเซกเมนต์ที่เลือกได้	ค่าออฟเซต
ค่า Fetch คำสั่ง	CS	–	IP
การทำ Stack	SS	–	SP
กำหนดตัวแปร	DS	CS, ES, SS	ค่า Address ที่กำหนด
สตริงต้นทาง	DS	CS, ES, SS	SI
สตริงปลายทาง	ES	–	DI
Base Register (BP)	SS	CS, ES, SS	ค่า Address ที่กำหนด

2.6 การระบุตำแหน่งในรีจีสเตอร์ (Addressing Data in Register)

การเข้าถึงข้อมูลในรีจิสเตอร์นับว่าเป็นหัวใจสำคัญในการเขียนโปรแกรมควบคุมการทำงานของไมโครโพรเซสเซอร์ 8086 สามารถ access เข้าไปในส่วนของโปรแกรม โดยใช้ค่าในรีจิสเตอร์ โดย **CS + IP –> physical address (20 bits)** แสดงได้ดังภาพที่ 2.10



ภาพที่ 2.10 แสดงการบวกค่าในรีจีสเตอร์ CS และ IP เพื่อคำนวณหา Physical Address
ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

ในการ Access เข้าไปยังข้อมูล เพื่อนำมาใช้ในการทำงาน เรียกว่า “addressing mode” ซึ่งในภาษาแอสแซมบลี ใช้คำสั่ง MOV: format –> MOV destination, source เมื่อ

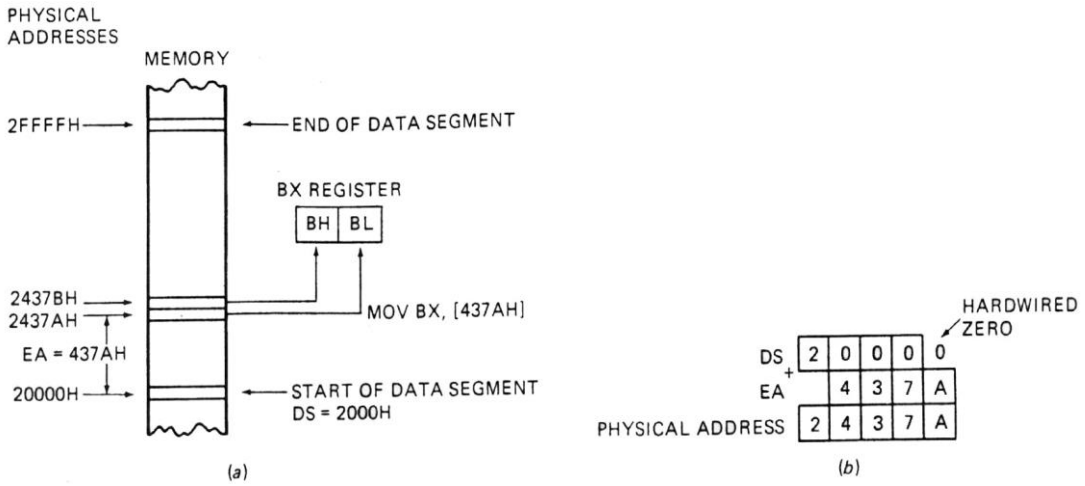
ชุดคำสั่งนี้ถูก execute 8086 จะทำการคัดลอก word หรือ byte จากตำแหน่งต้นทางไปยังจุดหมายปลายทาง Addressing Mode ใช้ในการระบุตำแหน่งของ operand ใน Memory ซึ่งสามารถแบ่งวิธีการเข้าถึงหน่วยความจำได้ดังนี้ คือ

2.6.1 Immediate Addressing Mode เป็นวิธีการอย่างง่ายในการกำหนดแอดเดรสในรีจิสเตอร์ กล่าวคือสามารถกำหนด operand ซึ่งเป็นค่าแอดเดรสให้กับชุดคำสั่งภาษาแอสแซมบลีได้โดยตรง ตัวอย่างเช่น ถ้าโปรแกรมต้องการที่จะใส่ค่า 437B (Hex) ลงใน CX Register ก็สามารถใช้คำสั่ง MOV CX, 437B (Hex) ได้โดยตรง

2.6.2 Register Addressing Mode ในกรณีที่ใช้ค่าในรีจิสเตอร์เป็น operand ของชุดคำสั่ง เช่น MOV CX, AX คือการคัดลอก เนื้อหาหรือข้อมูล(Content or Data) ที่อยู่ใน AX Register มาไว้ใน CX Register ใช้ในกรณีที่จำนวน bit เท่ากันเท่านั้น ในกรณีที่รีจิสเตอร์มีขนาดไม่เท่ากันจะไม่สามารถใช้คำสั่ง MOV CX, AL ได้ ทั้งนี้เนื่องจากหากใช้คำสั่งนี้ 8086 จะพยายามที่จะทำสำเนาข้อมูล (Copy Data) ในรูปแบบ byte-type (AL) เข้าไปเก็บในรีจิสเตอร์ซึ่งเป็นรูปแบบ word-type (CX) ซึ่งเป็นไปไม่ได้ อย่างไรก็ตามหากใช้คำสั่ง MOV AL, CX ก็ยังเป็นไปไม่ได้ เนื่องจากถึงแม้ว่าขนาดของรีจิสเตอร์จะสามารถรองรับข้อมูลได้ แต่ 8086 จะไม่ทราบว่าผู้เขียนโปรแกรมต้องการที่จะนำข้อมูลสำเนาไปเก็บไว้ในครึ่งใดของรีจิสเตอร์ CX ในกรณีเช่นนี้ โดยทั่วไป Assembler จะตรวจสอบพบ และจะบอกให้ทราบว่า Type error ดังนั้นการที่จะคัดลอกไบต์จาก AL ไปเก็บยังไบต์สูงของ CX จะใช้คำสั่ง MOV CH, AL และหากต้องการคัดลอกไบต์จาก AL ไปเก็บยังไบต์ต่ำของ CX จะใช้คำสั่ง MOV CL, AL

2.7 การเข้าถึงข้อมูลในหน่วยความจำ (Addressing Data in Memory)

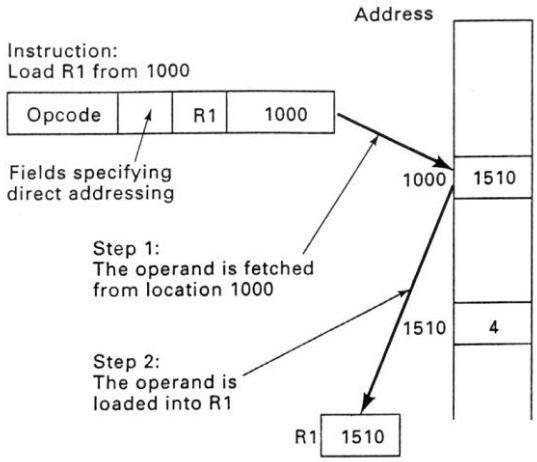
ที่ผ่านมาได้กล่าวถึง Addressing Mode ที่จะระบุตำแหน่งของ operand ในหน่วยความจำ ในกรณีที่ต้องการที่จะเข้าถึงข้อมูลในหน่วยความจำ 8086 จะต้องสร้าง physical address (20 bits) ซึ่งสามารถทำได้โดยการรวม Effective address (16 bits) เข้ากับ Segment Base Address (16 bits) แสดงได้ดังภาพที่ 2.11



ภาพที่ 2.11 แสดงการบวกค่าในรีจีสเตอร์ DS กับ EA เพื่อหา Physical Address
ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

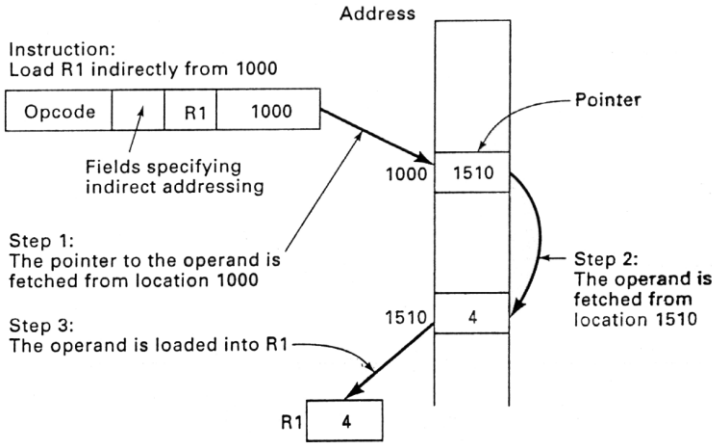
วิธีการเข้าถึงข้อมูลในหน่วยความจำในไมโครโพรเซสเซอร์ 8086 แบ่งได้เป็น 4 วิธี คือ

2.7.1 วิธีการเข้าถึงข้อมูลในหน่วยความจำโดยตรง (Direct Addressing Mode) เป็นวิธีการเข้าถึงข้อมูลในหน่วยความจำที่ง่ายที่สุด ซึ่งสามารถทำได้โดยการนำค่า effective address (16 bits) ใส่เข้าไปในชุดคำสั่งโดยตรง ตัวอย่างเช่นคำสั่ง MOV BL, [437A (Hex)] – 8086 จะคัดลอก Low byte –> Low address (437A) และ High byte –> High address (437B) ดังนั้นในกรณีที่ต้องการ Load รีจิสเตอร์ R1 ด้วยค่าที่อยู่ในแอดเดรส 1000 จะมีการทำงานแสดงได้ดังภาพที่ 2.12



ภาพที่ 2.12 แสดงวิธีการเข้าถึงข้อมูลในหน่วยความจำโดยตรง (Direct Addressing Mode)
ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

2.7.2 วิธีการเข้าถึงข้อมูลในหน่วยความจำโดยอ้อม (Indirect Addressing Mode) เป็นวิธีการเข้าถึงข้อมูลในหน่วยความจำที่แตกต่างจาก Direct Addressing ตรงที่แทนที่จะใส่ค่า effective address (16 bits) เข้าไปในชุดคำสั่งโดยตรง แต่จะใส่ค่า Address ของหน่วยความจำในตำแหน่งที่จัดเก็บ effective address เข้าไปในชุดคำสั่ง จะเห็นได้ว่าวิธีการนี้ค่าในแอดเดรสที่ระบุจะเป็นแอดเดรสที่บรรจุค่าที่ต้องการไว้ เราเรียกแอดเดรสภายในแอดเดรสที่ระบุว่า **"ตัวชี้ (Pointer)"** เช่นถ้าต้องการ Load รีจิสเตอร์ R1 ด้วยค่าในแอดเดรสที่แอดเดรส 1000 กำหนดไว้ จะมีการทำงาน แสดงได้ดังภาพที่ 2.13



ภาพที่ 2.13 แสดงวิธีการเข้าถึงข้อมูลในหน่วยความจำโดยอ้อม (Indirect Addressing Mode)
ที่มา: (http://www.school.net.th/library/snet1/hardware/z8086/cpu8086.html)

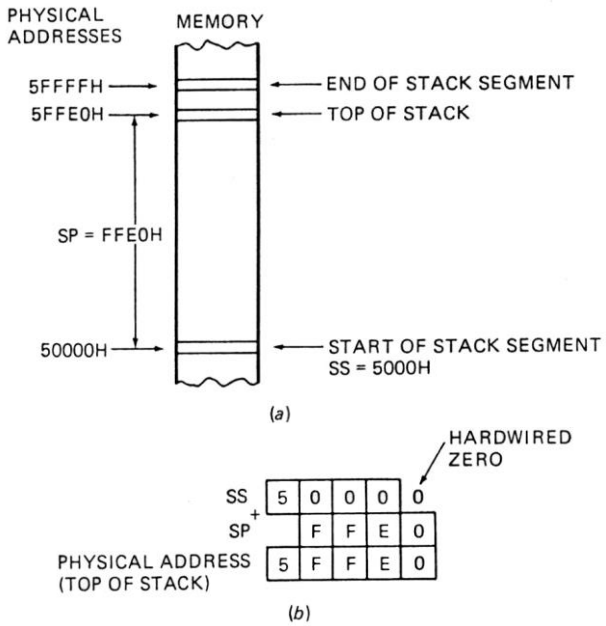
2.7.3 วิธีการเข้าถึงข้อมูลในหน่วยความจำโดยใช้ตัวชี้ (Indexing Addressing Mode) บางครั้งในการทำงานซึ่งต้องการกระทำบางอย่างกับชุดข้อมูลเป็นจำนวนมากที่มีโครงสร้างการจัดเก็บไว้เป็นลำดับ อาจจะสามารถทำได้โดยใช้วิธี Indirect Addressing แต่อย่างไรก็ตามหากต้องการ Load ค่า 2 ค่าจาก 2 แอดเดรส เข้ามาในรีจิสเตอร์ตัวเดียวกัน จำเป็นต้องใช้ Pointer 2 ตัว และทุกครั้งที่ Pointer เลื่อนไปชี้ที่แอดเดรสหนึ่ง Pointer จะถูกเพิ่มค่าขึ้น 1 เพื่อชี้ไปยังแอดเดรสถัดไป จึงสามารถทำงานกับชุดข้อมูลที่มีโครงสร้างการจัดเก็บเป็นลำดับได้ แต่อย่างไรก็ตามเนื่องจาก Pointer เป็นส่วนหนึ่งของข้อมูล ไม่ได้เป็นส่วนของโปรแกรม ดังนั้นผู้อื่นจะไม่สามารถทราบได้ว่าโปรแกรมนั้นต้องการใช้ทำอะไรแน่ การแก้ปัญหาอย่างหนึ่งคือการใช้รีจิสเตอร์เพิ่มขึ้นมาจากเดิมอีก 1 ตัว เรียกว่า "Index Register" โดยจะทำหน้าที่ในการจัดเก็บค่าดัชนีแอดเดรสของชุดข้อมูลเพื่อให้สะดวกต่อการเรียกใช้

ในปัจจุบันได้มีการพัฒนารีจิสเตอร์ชนิดนี้ให้มีขีดความสามารถในการเพิ่ม หรือลดค่าลงครั้งละ 1เพื่อให้สามารถจัดการกับชุดข้อมูลที่มีโครงสร้างการจัดเก็บเรียงเป็นลำดับได้โดยอัตโนมัติ จึงทำให้มีความสามารถในการ Address โดยอัตโนมัติเรียกว่า **“autoindexing”**

2.7.4 วิธีการเข้าถึงข้อมูลในหน่วยความจำโดยใช้สแต็ก (Stack Addressing Mode) เป็นวิธีการเข้าถึงข้อมูลในหน่วยความจำโดยใช้สแต็ก (Stack Addressing) ซึ่งสแต็กประกอบด้วยข้อมูลที่ถูกจัดเก็บเรียงเป็นลำดับไว้ในหน่วยความจำ โดยข้อมูลแรกจะถูกใส่ (Push) เข้าไปในสแต็ก และจะลงไปอยู่ที่ส่วนล่างสุดของสแต็ก ข้อมูลต่อๆ ไปก็จะถูกนำไปจัดเก็บในแอดเดรสถัดมา การที่จะเข้าไปเรียกข้อมูลในสแต็กจึงจำเป็นต้องมีแอดเดรสที่ระบุว่าเป็นส่วนบนสุดของสแต็ก ซึ่งเรียกว่า **“Stack Pointer”** กระบวนการเข้าไปยังสแต็กสามารถทำได้โดยนำค่า

SS (Stack Segment) + SP (Stack Pointer) –> Physical address (Stack’s top address)

แสดงได้ดังภาพที่ 2.14



ภาพที่ 2.14 แสดงวิธีการเข้าถึงข้อมูลในหน่วยความจำโดยใช้สแต็ก (Stack Addressing Mode)
ที่มา: (https://www.google.co.th/search?q= StacktAddressingMode&source)

สรุป

ไมโครโพรเซสเซอร์ (microprocessor) เป็นอุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งซึ่งมีลักษณะวงจรรวม(Integrated Circuit) หรือชิพ (Chip) โครงสร้างภายในจะมีวงจรรวมขนาดใหญ่ ส่วนไมโครคอนโทรลเลอร์ (microcontroller) เป็นชิพประมวลผลประเภทหนึ่งซึ่งทำหน้าตามโปรแกรมหรือขุดคำสั่งที่เขียนขึ้นมา ภายในประกอบด้วยวงจรรวมขนาดใหญ่ (Very Large Scale Integrated Circuit) การนำไมโครโพรเซสเซอร์ หลายๆ ตัวให้สามารถทำงานรวมกันได้ในเวลาเดียวกัน ทำให้ประสิทธิภาพการทำงานและความเร็วนั้นเพิ่มมากขึ้น

CPU 8086 ประกอบด้วยรีจิสเตอร์ที่เกี่ยวข้อง ออกเป็น 2 กลุ่ม คือ

1. กลุ่มข้อมูล (Data Group Register) แบ่งออกเป็น
   1.1 รีจีสเตอร์ทั่วไป (General–Purpose Register)
   1.2 รีจิสเตอร์ตัวชี้ (Instruction Pointer Register)
   1.3 แฟล็ก (flag)
2. กลุ่มกำหนดเซกเมนต์ (Segment Register) แบ่งออกเป็น
   2.1 รีจิสเตอร์กำหนดเซกเมนต์ (Segment Register)

คำถามทบทวน

1. จงอธิบายข้อแตกต่างที่ระหว่างไมโครโพรเซสเซอร์และไมโครคอนโทรลเลอร์
2. ไมโครโพรเซสเซอร์ จัดแบ่งออกตามลักษณะการใช้งานได้กี่ประเภทอะไรบ้าง
3. จงอธิบายโครงสร้างสถาปัตยกรรมของไมโครโพรเซสเซอร์ 8086 ประกอบด้วยอะไรบ้าง
4. จงอธิบายข้อแตกต่างที่สำคัญของไมโคโพรเซสเซอร์ 8086 และ 8088
5. ถ้า Code Segment (8086) เริ่มต้นที่ 90400H ค่าที่ถูกจัดเก็บในรีจีสเตอร์ CS คือค่าใด
6. ถ้า Code Segment (8086) เริ่มต้นที่ 90400H จงหา Physical Address ของหน่วยความจำที่จะถูก Fetch เข้ามาใช้งาน ถ้าในขณะนั้นรีจีสเตอร์ IP เก็บค่า 0345CH
7. ถ้า Stack Segment Register มีค่า 0500H และ Stack Pointer Register มีค่า 8225H จงหา Physical Address ส่วนบนสุดของสแต็ก (Top of Stack)
8. จงอธิบายข้อได้เปรียบของการใช้รีจีสเตอร์ในหน่วยประมวลผลกลาง (CPU) เป็นที่เก็บข้อมูลชั่วคราว แทนที่จะเก็บข้อมูลไว้ในหน่วยความจำ
9. จงอธิบายความแตกต่างระหว่างคำสั่ง MOV AX, 1234H กับ MOV AX, [1234H]

เอกสารอ้างอิง

ราชบัณฑิตยสถาน. (2544). *ศัพท์บัญญัติ ราชบัณฑิตยสถาน.* ค้นเมื่อ 15 ธันวาคม 2556,จาก: http://rirs3.royin.go.th/coinages/

รีจีสเตอร์ (2556). *วิกิพีเดีย สารานุกรมเสรี.* ค้นเมื่อ 15 ธันวาคม 2556, จาก: http://th.wikipedia.org/wiki/

ชูชัย ธนสารตั้งเจริญ, กำธร พานิชปฐมพงษ์. *ภาษาแอสแซมบลี 80286/80386(PC).* กรุงเทพฯ :สำนักพิมพ์ซีเอ็ดยูเคชั่น บมจ, 2536.

ธีรวัฒน์ ประกอบผล. *ระบบคอมพิวเตอร์และภาษาแอสแซมบลี.* กรุงเทพฯ :สำนักพิมพ์ส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2537.