บทที่ 5

ICE ในระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ด

5.1 ICE ระดับบอร์ด

ระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดที่สร้างขึ้น เป็นระบบพัฒนาไมโครโปรเซสเซอร์หรือ MDS ที่ใช้งานร่วมกับเครื่องไมโครคอมพิวเตอร์ IBM PC จากที่กล่าวมาแล้วในบทที่ 2 ว่า ระบบพัฒนาไมโครโปรเซสเซอร์ประกอบด้วยส่วนที่เป็นฮาร์ดแวร์และส่วนที่เป็นซอฟต์แวร์ ส่วนที่เป็นฮาร์ดแวร์ของระบบพัฒนาไมโครโปรเซสเซอร์ที่สร้างขึ้นนี้ ได้แก่ ซีพียู และหน่วยความจำหลักที่อยู่บนบอร์ดซีพียู หน่วยเก็บข้อมูลและ CRT คอนโซลซึ่งอาศัยดิสค์ไดรฟ์ คีย์บอร์ดและจอภาพของเครื่องไมโครคอมพิวเตอร์ IBM PC เครื่องโปรแกรมอีพรอมซึ่งใช้เครื่องโปรแกรม UNIPRO และอุปกรณ์ช่วยในการดีบัก ICE ซึ่งเป็น ICE ระดับบอร์ดที่ใช้ดีบักร่วมกับเครื่องไมโครคอมพิวเตอร์IBM PC ผ่านทางพอร์ต RS-232C นั่นเอง ส่วนซอฟต์แวร์ของระบบพัฒนาไมโครโปรเซสเซอร์ในงานวิจัยครั้งนี้ได้แก่ โปรแกรมมอนิเตอร์ที่อยู่บน ICE ระดับบอร์ด โปรแกรมแอสเซมเบลอร์ของ8085และ Z-80 โปรแกรมคอนไพล์เลอร์ของภาษา C และ PL/M โปรแกรมลิงค์เกอร์โหลดเดอร์และดีบักเกอร์ต่าง ๆ ซึ่งซอฟต์แวร์ทั้งหมดนี้อยู่บนเครื่องไมโครคอมพิวเตอร์ IBM PC

ICE ในระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดนี้ เป็น ICE ระดับบอร์ดแบบซีพียูเดียวซึ่งยังไม่มีใครทำขึ้นมาก่อน ICE ระดับบอร์ดนี้จะถูกออกแบบให้อยู่บนบอร์ดที่เรียกว่าบอร์ดอีมูเลชัน (Emulation Board) ซึ่งเป็นบอร์ดประกบกับด้านบนของบอร์ดซีพียู ไม่ได้เป็นบอร์ดที่เสียบเข้ากับแผ่นสัญญาณด้านหลัง การเชื่อมต่อสัญญาณข้อมูลสัญญาณแอดเดรสและสัญญาณควบคุมต่าง ๆ จึงไม่ได้ผ่านทาง STD บัสโดยตรง แต่ผ่านทาง Jumper ที่ใช้ประกบกับบอร์ดซีพียูนั่นเอง ความสามารถในการดีบักของบอร์ดอีมูเลชันในระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ด มีดังต่อไปนี้

1. การจัดการกับหน่วยความจำ (Memory Management) สามารถเรียกดูค่าในหน่วยความจำ (Display User Memory) แก้ไขข้อมูลในหน่วยความจำ (Enter User Memory) และป้อนข้อมูลเป็นกลุ่มได้ (Fill Block)

2. การจัดการกับรีจีสเตอร์ (Register Management) สามารถเรียกดูและแก้ไขค่าในรีจีสเตอร์ได้ (Register Examine)

3. การแสดงข้อมูลในหน่วยความจำเป็นภาษาแอสเซมบลี (Unassemble)

4. การป้อนโปรแกรมเป็นภาษาแอสเซมบลีทีละบรรทัด (Line Assemble)

5. การจัดการกับพอร์ตไอ/โอ ได้แก่ การอ่านและเขียนค่ากับพอร์ต (In port /Out port)

6. การทำงานทีละคำสั่ง (Single Step หรือ Trace) ซึ่งเรียกดูค่าต่าง ๆได้

7. การสั่งให้ปฏิบัติงานในตำแหน่งหน่วยความจำปัจจุบัน (Execute Program from Current Address) คือตำแหน่งที่ PC (Program Counter) ชี้อยู่หรือสั่งให้ทำงานในตำแหน่งที่กำหนดต่อท้ายคำสั่งก็ได้

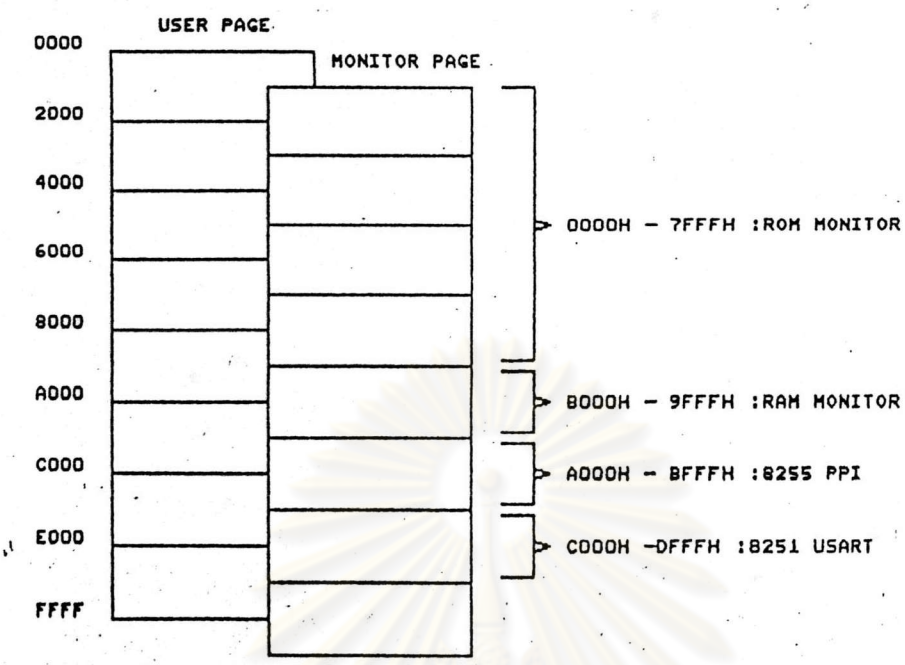
8. การกำหนดจุดหยุดปฏิบัติงาน(Breakpoint) และการยกเลิกจุดหยุดปฏิบัติงาน (Clear Breakpoint)

9. การถ่ายข้อมูลจากเครื่องไมโครคอมพิวเตอร์ IBM PC ลงหน่วยความจำของระบบ (Load File หรือ Download)

10. การเก็บข้อมูลจากหน่วยความจำของระบบลงดิสค์ (Write to File หรือ Upload)

11. การหยุดรันโปรแกรมผู้ใช้ด้วยการเคาะแป้นพิมพ์ของเครื่องไมโครคอมพิวเตอร์ IBM PC หรือการกดปุ่มมอนิเตอร์บนบอร์ดอีมูเลชัน

จากการที่บอร์ดอีมูเลชันเป็น ICE แบบซีพียูเดียว ระบบจึงมีซีพียูเพียวตัวเดียวซึ่งอยู่บนบอร์ดซีพียูและหน่วยความจำ ส่วนบนบอร์ดอีมูเลชันนั้นไม่มีตัวซีพียูอยู่ ฉะนั้นซีพียูของระบบตัวนี้จะต้องทำหน้าที่เป็นทั้ง Target Processor และ Control Processor ที่ต้องรันโปรแกรมทั้งของผู้ใช้และโปรแกรมมอนิเตอร์ซึ่งเป็นโปรแกรมควบคุมในการดีบัก จากที่กล่าวมาแล้วว่าได้ออกแบบให้ผู้ใช้สามารถใช้หน่วยความจำได้ 64 K (0000H-FFFFH) เต็มความสามารถของซีพียูขนาด 8 บิต การอ้างแอดเดรสหน่วยความจำมอนิเตอร์จึงต้องมีลักษณะเหมือนเป็นแอดเดรสของหน่วยความจำอีกเพจซ้อนอยู่ นอกจากนี้ผู้ออกแบบยังเปิดโอกาสให้ผู้ใช้สามารถอ้างแอดเดรสของพอร์ตไอ/โอได้เต็ม 256 พอร์ต (00H-FFH) ด้วย ดังนั้นการอ้างแอดเดรสพอร์ตไอ/โอของมอนิเตอร์จึงถูกออกแบบให้มีการอ้างแอดเดรสแบบหน่วยความจำแทนโดยจะอยู่ในเพจเดียวกับหน่วยความจำมอนิเตอร์ ดังแสดงแอดเดรสต่าง ๆ ของเพจมอนิเตอร์ได้ในรูปที่ 5.1 และจากการออกแบบให้มีแอดเดรสหน่วยความจำ 2 เพจซ้อนกันอยู่ทำให้ต้องมีวงจรควบคุมการสลับเพจ เพื่อให้ซีพียูทำงานตามโปรแกรมในเพจใดเพจหนึ่ง ซึ่งอาจเป็นเพจผู้ใช้หรือเพจมอนิเตอร์



รูปที่ 5.1 แสดงการอ้างแอดเดรสของมอนิเตอร์

5.2 เทคนิคการสลับเพจ

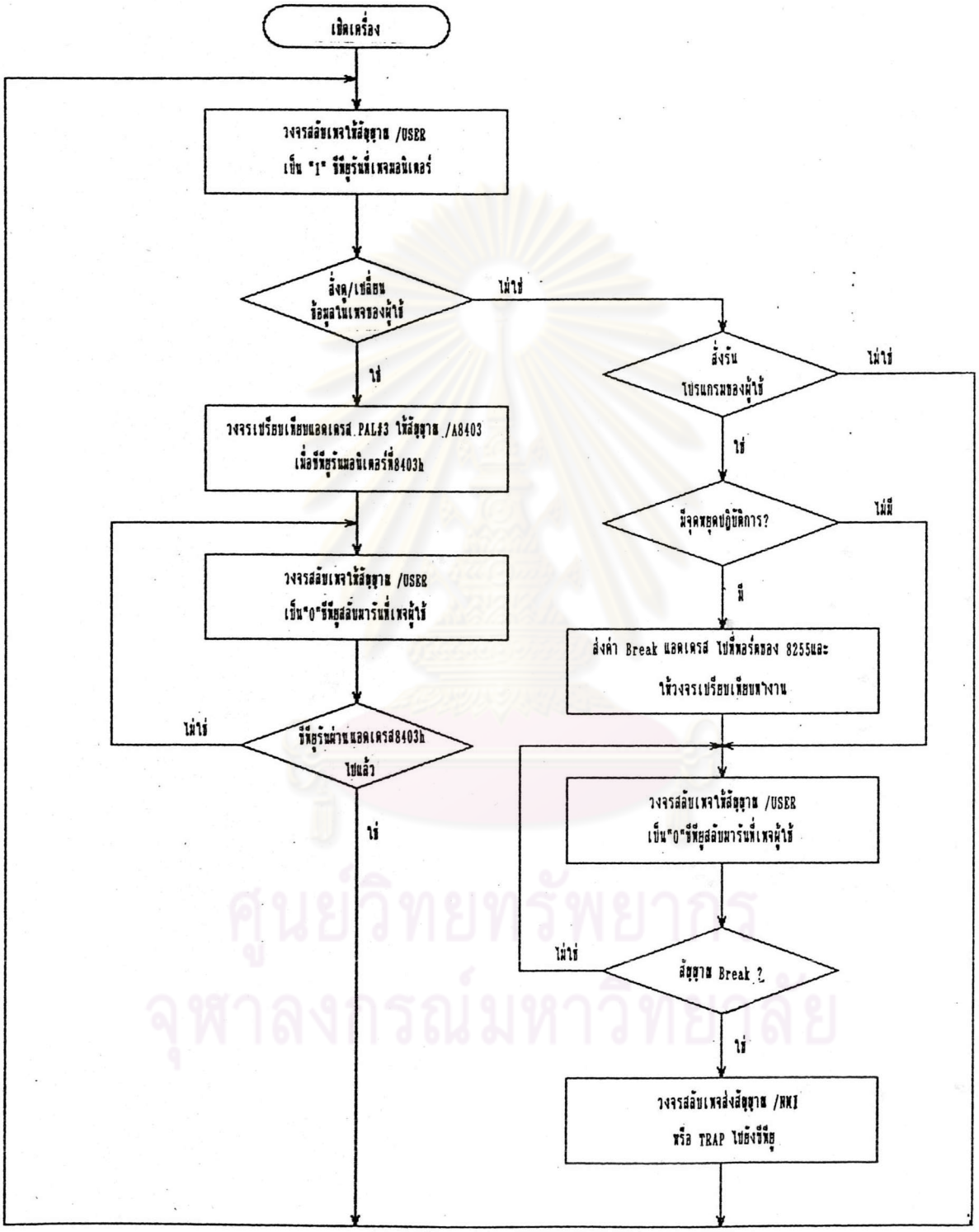
ในขบวนการดีบักที่ใช้บอร์ดอีมูเลชัน เริ่มแรกเมื่อเปิดเครื่องขึ้นมาซีพียูจะต้องรันที่เพจมอนิเตอร์เสมอและจะสลับเพจไปสู่เพจผู้ใช้ได้ก็ต่อเมื่อมีการอ่านเขียน หรือสั่งรันโปรแกรมของผู้ใช้ ส่วนการอ่านเขียนข้อมูลกับพอร์ตไอ/โอของผู้ใช้ไม่ต้องมีการสลับเพจเนื่องจากเป็นการอ้างแอดเดรสแบบไอ/โอ จากความสามารถในการดีบักของบอร์ดอีมูเลชันคำสั่งที่เกี่ยวข้องกับการสลับเพจมีดังนี้

1. คำสั่งอ่าน/เขียนข้อมูลในหน่วยความจำผู้ใช้
2. คำสั่งรันโปรแกรมของผู้ใช้
3. คำสั่งรันโปรแกรมของผู้ใช้ที่มี Breakpoint และคำสั่ง Single Step

5.2.1 การอ่าน/เขียนข้อมูลในหน่วยความจำของผู้ใช้

เริ่มแรกซีพียูจะทำงานอยู่ในเพจมอนิเตอร์ หลังจากได้รับคำสั่งขอดูหรือแก้ไขค่าในหน่วยความจำของผู้ใช้ถูกต้องแล้ว ซีพียูจะกระโดดจากโปรแกรมมอนิเตอร์หลักมาสู่มอนิเตอร์รูทีนย่อยของการอ่านเขียนข้อมูลหน่วยความจำผู้ใช้ซึ่งรูทีนย่อยนี้จะสั่งให้ไปเขียนคำสั่งโหลดค่าจากหน่วยความจำที่แอดเดรสซึ่งรีจีสเตอร์ BC ชี้อยู่มาไว้ที่รีจีสเตอร์ A ณ แอดเดรส 8403H ของหน่วยความจำมอนิเตอร์ในกรณีที่ผู้ใช้ต้องการอ่านข้อมูลจากหน่วยความจำ หรือสั่งให้ไปเขียนคำสั่งเก็บค่าในรีจีสเตอร์ A ลงสู่หน่วยความจำที่แอดเดรสซึ่งรีจีสเตอร์ BC ชี้อยู่

ในกรณีที่ผู้ใช้ต้องเขียนข้อมูลเข้าหน่วยความจำ เช่น ผู้สั่งผ่านแป้นพิมพ์ของเครื่องไมโครคอมพิวเตอร์ว่าต้องการอ่านข้อมูลจากหน่วยความจำที่แอดเดรส 1800H มอนิเตอร์รูทีนย่อยก็จะสั่งให้เขียนคำสั่ง LD A,(BC) ไปไว้ที่แอดเดรส 8403H ของเพจมอนิเตอร์ และเตรียมการให้รีจีสเตอร์ BC เท่ากับ 1800H เมื่อซีพียูรันรูทีนย่อยมาถึงแอดเดรส 8403H ของเพจมอนิเตอร์ ซีพียูจะเฟทช์คำสั่งโหลดนี้ขณะเดียวกันฮาร์ดแวร์ส่วนหนึ่งของส่วนควบคุมการสลับเพจ (PAL#3) ก็จะส่งสัญญาณ /A8403 ไปให้วงจรควบคุมการสลับเพจ (PAL#5) เพื่อส่งสัญญาณ /USER ที่เป็น "0" ไปให้หน่วยความจำของผู้ใช้แอกทีฟในจังหวะที่ 2 ของคำสั่งโหลด ซึ่งเป็นจังหวะที่ซีพียูอ่านข้อมูลจากหน่วยความจำพอดี ดังนั้นซีพียูจะอ่านข้อมูลจากหน่วยความจำที่แอดเดรส 1800H ซึ่งเป็นข้อมูลในหน่วยความจำของผู้ใช้ และเมื่อแอดเดรสเปลี่ยนไปวงจร PAL#3 ก็เลิกส่งสัญญาณ /A8403 ทำให้วงจรควบคุมการสลับเพจ PAL#5 ให้สัญญาณ /USER เป็น"1" อีกครั้ง หน่วยความจำมอนิเตอร์จึงแอกทีฟแทนหน่วยความจำของผู้ใช้ ซีพียูก็จะกลับมาเฟทช์คำสั่งต่อที่เพจมอนิเตอร์นั่นเอง จะเห็นว่าเป็นการสลับเพจจากเพจมอนิเตอร์ไปยังเพจผู้ใช้เพียงชั่วขณะแล้วกลับมาทำงานที่เพจมอนิเตอร์ต่อไป ดังแสดงในรูปที่ 5.2



รูปที่ 5.2 แสดงแผนภูมิการทำงานของส่วนควบคุมการสลับเพจ

5.2.2 การสั่งรันโปรแกรมของผู้ใช้และการหยุดรันโปรแกรม

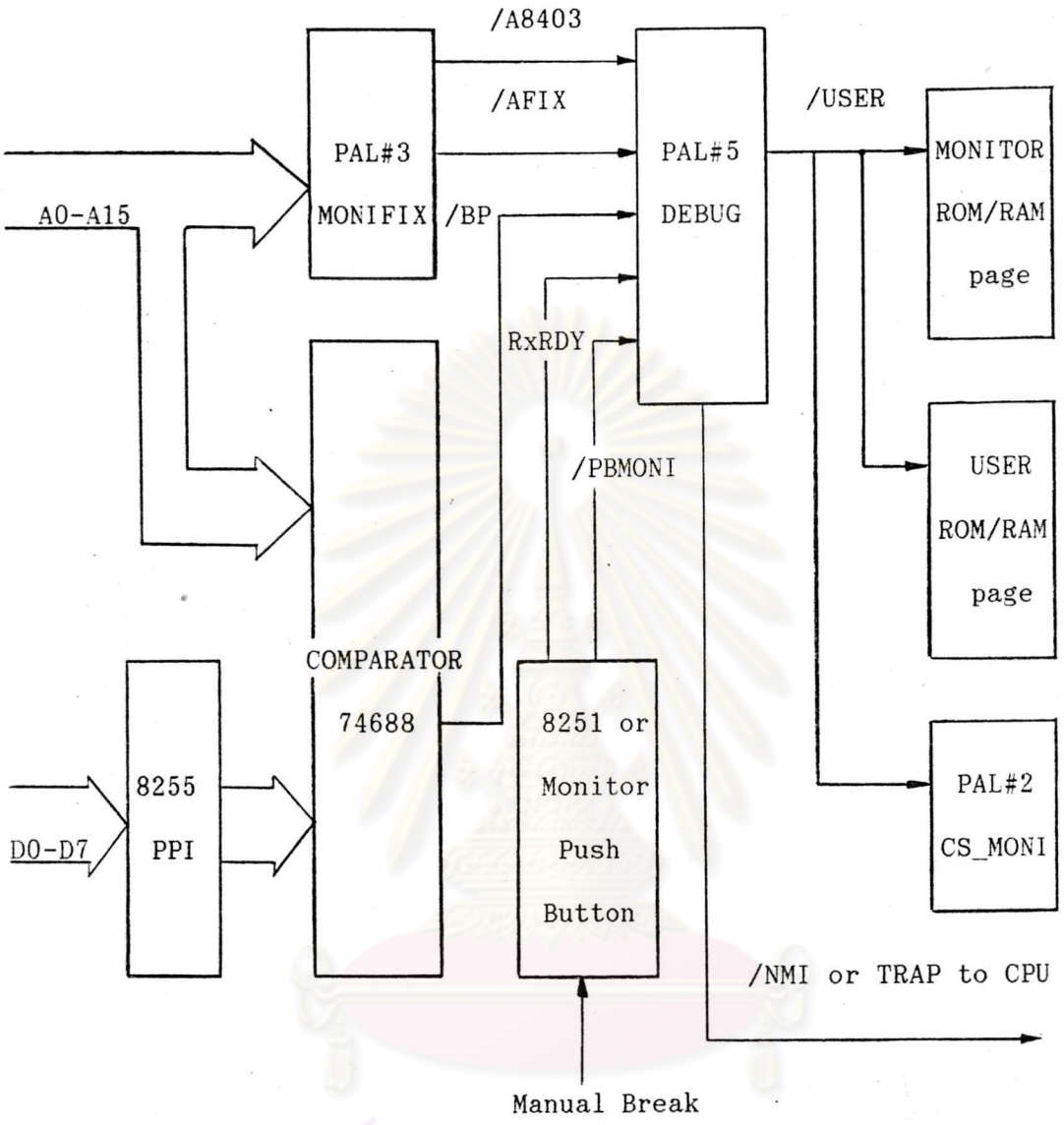
หลังจากผู้ใช้ป้อนโปรแกรมลงสู่หน่วยความจำแล้วต้องการทดสอบโปรแกรมที่เขียนขึ้นว่าใช้งานได้จริงหรือไม่ ก็จะสั่งรันโปรแกรมซึ่งมีอยู่ 2 รูปแบบ คือเริ่มรันโปรแกรมจากตำแหน่ง PC ปัจจุบัน หรือเริ่มรันที่ตำแหน่งที่กำหนดต่อท้ายคำสั่ง ในกรณีหลังนี้จะมีการนำเอาค่าที่กำหนดมาใส่เป็นค่า PC ของผู้ใช้ก่อนที่จะสั่งรันปกตินั่นเอง นอกจากนี้ในการดีบักโปรแกรมมักจะสั่งรันเป็นช่วงได้เพื่อติดตามตรวจหาความผิดพลาดของโปรแกรม ดังนั้นบอร์ดอีมูเลชันจึงมีคำสั่งตั้งจุดหยุดปฏิบัติการและคำสั่งยกเลิกจุดหยุดปฏิบัติการด้วย

ขั้นตอนการทำงานของคำสั่งรันโปรแกรม จะเริ่มจากซีพียูทำงานอยู่ที่เพจมอนิเตอร์ได้รับคำสั่งรันโปรแกรมของผู้ใช้ ซีพียูจะมาตรวจว่ามีการตั้งจุดหยุดปฏิบัติการหรือไม่ ถ้ามีก็จะส่งค่าแอดเดรสของจุดหยุดปฏิบัติการไปที่พอร์ตของ 8255 เพื่อให้วงจรเปรียบเทียบแอดเดรสทำการเปรียบเทียบกับค่าในแอดเดรสบัส ก่อนกระโดดไปที่มอนิเตอร์รูทีนย่อยของการรันโปรแกรมผู้ใช้ซึ่งจะสั่งให้เขียนคำสั่ง Jump เพื่อตั้งค่า PC เป็นค่าแอดเดรสที่ต้องการรัน ณ แอดเดรสที่ถูกกำหนดให้ทำงานสัมพันธ์กับวงจรควบคุมการสลับเพจ ก็คือ ที่แอดเดรส 8401H และเนื่องจากทั้งโปรแกรมมอนิเตอร์และโปรแกรมผู้ใช้ต้องใช้รีจีสเตอร์ชุดเดียวกัน เพราะเป็น ICE แบบซีพียูเดียว จึงต้องมีการเก็บบันทึกค่ารีจีสเตอร์และค่าสถานะต่าง ๆ ที่จำเป็นของโปรแกรมผู้ใช้ไว้ก่อนแล้วและต้องมีการคืนค่าต่าง ๆ กลับสู่รีจีสเตอร์และระบบก่อนจะสั่งให้เริ่มทำงานตามโปรแกรมในเพจของผู้ใช้ดังนั้นหลังจากมอนิเตอร์รูทีนย่อยนี้เขียนคำสั่งJump PCผู้ใช้ที่แอดเดรส 8401H และคืนค่าเดิมของรีจีสเตอร์ผู้ใช้แล้ว ซีพียูจะทำการเฟทช์คำสั่งที่แอดเดรส 8401H ซึ่งมีคำสั่ง Jump เป็นคำสั่งสุดท้ายของเพจมอนิเตอร์ก่อนจะสลับไปเพจผู้ใช้ หลังจากจังหวะเฟทช์ที่แอดเดรส 8401H ก็จะทำการอ่านค่าแอดเดรสที่ต้องการจากแอดเดรส 8402H และ 8403H เมื่อค่าในแอดเดรสบัสมีค่า 8403H วงจรสลับเพจจะได้รับสัญญาณ /A8403 จาก PAL#3 ดังรูปที่ 5.3 และทำการสลับเพจไปที่เพจผู้ใช้โดยให้สัญญาณ /USER เป็น"0" ดังนั้นคำสั่งที่ซีพียูจะเฟทช์ต่อไปก็คือคำสั่งในโปรแกรมของผู้ใช้ที่แอดเดรสที่ต้องการซึ่งมี PC ชี้อยู่และทำงานที่เพจผู้ใช้ไปจนกว่าจะเจอการ Break

การหยุดทำงานตามโปรแกรมของผู้ใช้เกิดขึ้นได้หลายกรณีคือ อาจเกิดจากการตั้งค่า Breakpointไว้เพื่อให้ซีพียูหยุดรันโปรแกรมของผู้ใช้ ณ ตำแหน่งที่กำหนด หรืออาจเกิดจากการหยุดรันโปรแกรมผู้ใช้อย่างกระทันหัน เมื่อผู้ใช้เคาะแป้นพิมพ์ของเครื่องไมโครคอมพิวเตอร์ IBMPC หรือกดปุ่มมอนิเตอร์บนบอร์ดอีมูเลชันขณะซีพียูกำลังรันโปรแกรมของผู้ใช้ หรืออาจเกิดจากการสั่งให้ทำงานแบบ Single Step ซึ่งการสั่งรันแบบนี้จะใช้วิธีเดียวกับการสั่งรันแบบมี Breakpoint คือ ให้เอาค่าแอดเดรสที่สั่งรันไปเป็นแอดเดรสของ Breakpoint

ดังนั้นหลังจากซีพียูรันไปหนึ่งคำสั่งก็จะพบการสั่งหยุดรันโปรแกรมผู้ใช้ทันที ขั้นตอนการหยุดรันโปรแกรมผู้ใช้ในกรณีของ Breakpoint หรือ Single Step จะเริ่มจากวงจรเปรียบเทียบแอดเดรสได้รันแอดเดรสที่กำหนดเป็นจุดหยุดจากพอร์ตของ 8255 ดังรูปที่ 5.3 จากนั้นซีพียูสลับไปรันที่โปรแกรมของผู้ใช้ตามรูทีนย่อยของคำสั่งรันโปรแกรมผู้ใช้ วงจรเปรียบเทียบจะเทียบค่าในแอดเดรสบัสกับค่าที่กำหนด หากมีแอดเดรสตรงกันก็จะส่งสัญญาณ /BP เข้าสู่วงจรควบคุมการสลับเพจ PAL#5 ซึ่งจะทำการส่งสัญญาณ /NMI หรือสัญญาณ TRAP ไปสู่ซีพียูเพื่อให้ซีพียูทำคำสั่งที่กำลังรันจนจบแล้วเก็บค่า PC ถัดไปลงในสแต็กของผู้ใช้ ก่อนกระโดดไปทำรูทีนของสัญญาณ /NMI ที่แอดเดรส 0066H สำหรับซีพียูเบอร์ Z-80 หรือรูทีนของสัญญาณ TRAP ที่แอดเดรส 0024H สำหรับซีพียูเบอร์ 8085 และขณะนั้นเองวงจร PAL#3 จะตรวจเจอว่ามีการเฟทช์คำสั่งที่แอดเดรสเหล่านี้ก็จะให้สัญญาณ /AFIXไปยังวงจรควบคุมการสลับเพจ PAL#5 เพื่อให้ส่งสัญญาณ /USER เป็น "1" ออกมาและมีผลทำให้ซีพียูกลับมาทำงานที่เพจมอนิเตอร์ รวมทั้งเก็บค่า PC ถัดไปของโปรแกรมผู้ใช้ซึ่งถูกพุชลงสแต็กของผู้ใช้ด้วย โดยใช้วิธีการเขียนคำสั่ง POP ไปที่แอดเดรส 8403H ของมอนิเตอร์แล้วทำขบวนการสลับเพจ เหมือนการอ่านค่าจากหน่วยความจำผู้ใช้เพื่อเอาค่า PC เดิมมา จากนั้นจึงกลับไปเริ่มต้นทำงานที่โปรแกรมหลักของมอนิเตอร์รอรับคำสั่งจากผู้ใช้ต่อไป

ส่วนขั้นตอนการหยุดรันโปรแกรมผู้ใช้ ในกรณีของการเคาะแป้นพิมพ์ของเครื่องไมโครคอมพิวเตอร์ IBM PC หรือกดปุ่มมอนิเตอร์ที่อยู่บนบอร์ดอีมูเลชันขณะซีพียูกำลังรันโปรแกรมผู้ใช้นั้นจะมีขั้นตอนดังนี้คือ ขณะซีพียูกำลังทำงานอยู่ในเพจผู้ใช้หากมีการเคาะแป้นพิมพ์หรือกดปุ่มมอนิเตอร์ จะทำให้วงจรรับส่งข้อมูลแบบอนุกรม 8251 ส่งสัญญาณ RxRDY เข้าสู่วงจรควบคุมการสลับเพจ PAL#5 หรือวงจรควบคุมการกดปุ่มมอนิเตอร์ส่งสัญญาณ /PBMONI เข้าสู่วงจรควบคุมการสลับเพจตามลำดับซึ่งจะมีผลทำให้วงจรควบคุมการสลับเพจ PAL#5 ส่งสัญญาณ /NMI หรือสัญญาณ TRAP สู่ซีพียูและมีขบวนการทำงานอย่างเดียวกับกรณีของการตั้งค่า Breakpoint คือ สลับกลับมาทำงานที่เพจมอนิเตอร์ และเก็บค่ารีจีสเตอร์ของผู้ใช้ลงในหน่วยความจำมอนิเตอร์แล้วจึงกลับไปเริ่มทำงานที่โปรแกรมหลัก



รูปที่ 5.3 แสดงส่วนประกอบของวงจรควบคุมการสลับเพจ

5.3 โปรแกรมควบคุมการดีบัก

โปรแกรมควบคุมการดีบัก เป็นโปรแกรมควบคุมระบบพัฒนาขณะทำการดีบักโปรแกรมผู้ใช้โดยใช้บอร์ดอีมูเลชันร่วมกับเครื่องไมโครคอมพิวเตอร์ IBM PC หรือเทอร์มินัล VT-100 การดีบักโปรแกรมโดยใช้บอร์ดอีมูเลชันร่วมกับเทอร์มินัลนั้น โปรแกรมควบคุมการดีบักทั้งหมดต้องอยู่บนหน่วยความจำมอนิเตอร์ของระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ด เพราะเราต้องส่งข้อมูลทุกอย่างแม้กระทั่งลักษณะข้อความที่ปรากฏบนจอภาพให้กับเทอร์มินัล ส่วนการดีบักโปรแกรมที่ใช้บอร์ดอีมูเลชันร่วมกับเครื่องไมโครคอมพิวเตอร์ IBM PC นั้น เราอาจแบ่งโปรแกรมควบคุมการดีบักให้ส่วนหนึ่งอยู่บนเครื่อง IBM PC และอีกส่วนอยู่บนหน่วยความจำ

มอนิเตอร์ของบอร์ดอีมูเลชัน หรือโปรแกรมทั้งหมดอยู่บนหน่วยความจำมอนิเตอร์ก็ได้ ทั้งนี้เนื่องจากเครื่องไมโครคอมพิวเตอร์ IBM PC มีความสามารถในการเก็บข้อมูลและประมวลผลในตัวจึงสามารถรับข้อมูลดิบจากโปรแกรมมอนิเตอร์มาจัดแสดง หรือเปลี่ยนแปลงรูปแบบข้อมูลก่อนขึ้นจอภาพได้ซึ่งอาจจะเป็นรูปแบบของ Pop-up Menu หรือ Pull-down Menu เพื่อให้ใช้งานได้สะดวกยิ่งขึ้น นอกจากนี้เครื่องไมโครคอมพิวเตอร์ IBM PC ยังสามารถจำลองตัวเองเป็นเทอร์มินัลด้วยซอฟต์แวร์สนับสนุนต่าง ๆ เช่น โปรแกรม PROCOMM หรือโปรแกรม X-Talk เป็นต้น จากการที่งานวิจัยครั้งนี้ได้ออกแบบให้บอร์ดอีมูเลชันใช้ได้ทั้งกับเครื่องไมโครคอมพิวเตอร์ IBM PC และเทอร์มินัล VT-100 ทำให้เราสามารถแบ่งโปรแกรมควบคุมการดีบักออกเป็น 2 ส่วนคือ ส่วนที่อยู่บนเครื่องไมโครคอมพิวเตอร์ IBM PC ซึ่งเป็นโปรแกรมจัดการทำงานแบบเมนูและส่วนที่บนบอร์ดอีมูเลชัน ซึ่งเรียกว่า โปรแกรมมอนิเตอร์ ในส่วนแรกนั้นไม่ได้อยู่ในขอบเขตของงานวิจัยครั้งนี้ ส่วนโปรแกรมมอนิเตอร์บนบอร์ดอีมูเลชันจะถูกออกแบบเพื่อไว้สำหรับการทำงานแบบเมนูบนเครื่อง IBM PC ด้วย เราสามารถแบ่งรูปแบบการทำงานของคำสั่งในการดีบักออกเป็น 2 โหมด คือ

1. โหมดเทอร์มินัล (Terminal Mode) เป็นคำสั่งที่ใช้ทำงานร่วมกับอุปกรณ์แบบเทอร์มินัล ซึ่งเป็นแบบที่โปรแกรมมอนิเตอร์ต้องส่งรายละเอียดทุกอย่างแม้แต่รูปแบบของการขึ้นหน้าจอภาพไปให้ คำสั่งในโหมดนี้จึงสามารถใช้ได้ทั้งกับเทอร์มินัล VT-100 และเครื่องไมโครคอมพิวเตอร์ IBM PC ซึ่งตัวมันเองสามารถเลียนแบบการทำงานของเทอร์มินัลได้ด้วยซอฟต์แวร์สนับสนุนต่าง ๆ เช่น โปรแกรม PROCOMM เป็นต้น

2. โหมดโฮสต์(Host Mode) เป็นคำสั่งที่ใช้เฉพาะกับเครื่องไมโครคอมพิวเตอร์ IBM PC ในกรณีที่ทำงานแบบเมนู โดยตัว IBM PC ต้องมีโปรแกรมจัดการกับข้อมูลดิบที่โปรแกรมมอนิเตอร์ส่งไปให้เพื่อจัดแสดงข้อมูลในจอเมนู

เราสามารถแสดงรูปแบบของคำสั่งที่ใช้ในการดีบักทั้งโหมดเทอร์มินัลและโหมดโฮสต์ดังตารางที่ 5.1 และ ตารางที่ 5.2 ตามลำดับ

ตารางที่ 5.1 รูปแบบของคำสั่งในโหมดเทอร์มินัล

คำสั่ง	รูปแบบของคำสั่ง	หมายเหตุ
A - Assemble One Line	>A xxxx	
B - Breakpoint	>B xxxx	Set Break
	>B	View Break
C - Clear Breakpoint	>C	
D - Display User Memory	>D xxxx	
	>D xxxx yyyy	
E - Edit User Memory	>E xxxx[nn]	
F - Fill Block User Memory	>F xxxx yyyy nn	
G - GO	>G	Run from PC
	>G xxxx	Run from xxxx
H - Help Menu	>H	
I - In Port	>I pp	
O - Out Port	>O pp nn	
R - Register Examine	>R	View Register
	>R r	Edit Register
T - Trace (Single Step)	>T	
U - Unassemble	>U [xxxx yyyy]	
X - Reset	>X	
L - Load File (Intel Hex)	>L	on IBM PC only
W - Write to File	>W xxxx yyyy	on IBM PC only

xxxx : Start Address

yyyy : End Address

nn : Data

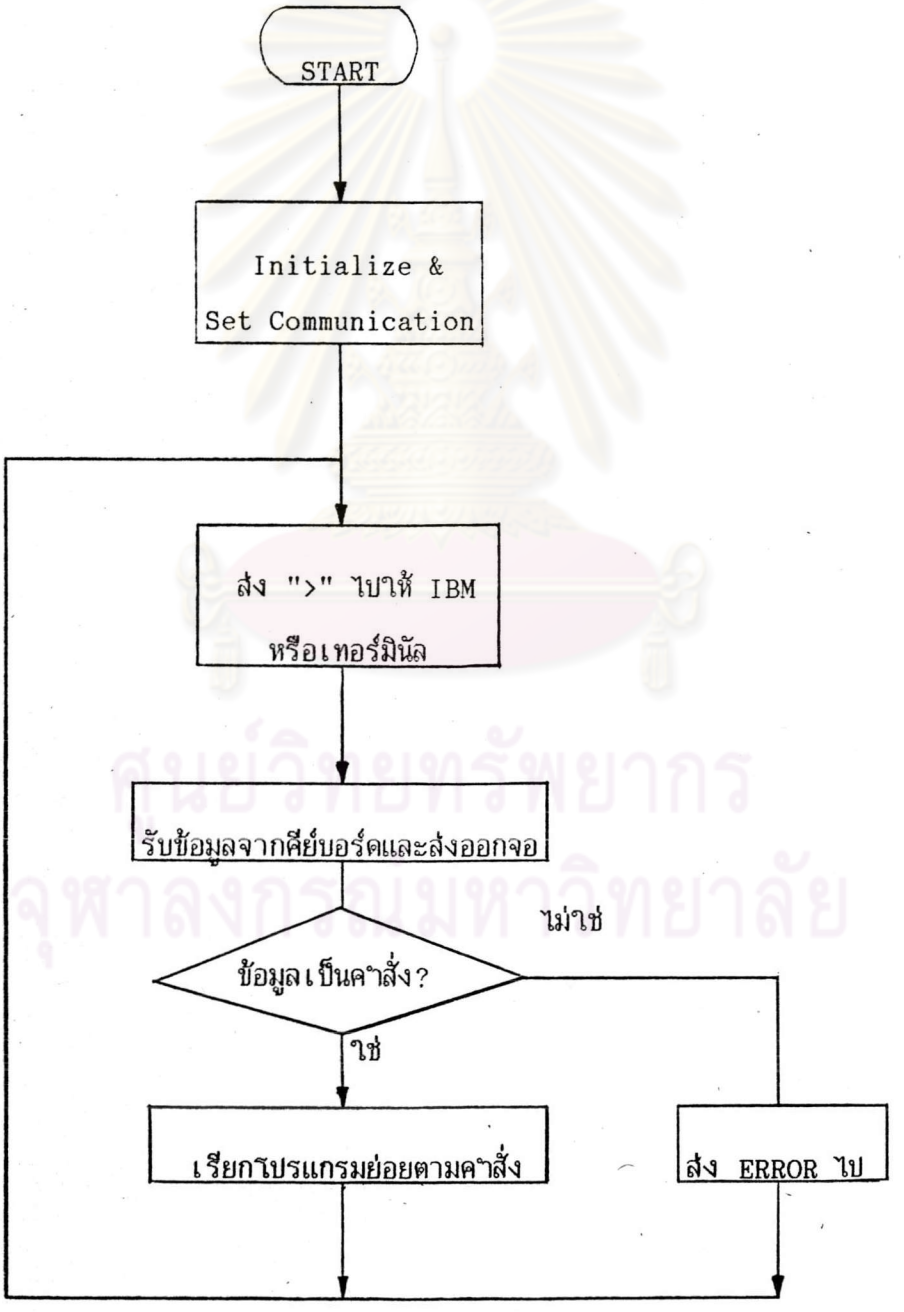
pp : Port I/O

r : Register Name

ตารางที่ 5.2 รูปแบบของคำสั่งในโหมดโฮสที่ใช้เฉพาะกับเมนูบน IBM PC

คำสั่ง	รูปแบบของคำสั่ง
J - Display User Memory	-เมนูเป็นผู้รับคำสั่งจากผู้ใช้ในรูปแบบของเมนู ไม่ว่าจะเป็นคำสั่ง"D","U"หรือ"W" เมนูจะส่ง"J"มาให้กับมอนิเตอร์ พร้อมด้วยแอดเดรสเริ่มต้นและจำนวนไบต์ เพื่อให้มอนิเตอร์ส่งค่าในหน่วยความจำไปให้เมนูจัดการแสดงผลตามคำสั่งของผู้ใช้
K - Enter User Memory	-เมนูเป็นผู้รับคำสั่งจากผู้ใช้ในรูปแบบของเมนู ไม่ว่าจะเป็นคำสั่ง"A","E","F"หรือ"L" เมนูจะส่ง"K" มาให้กับมอนิเตอร์พร้อมด้วยแอดเดรสเริ่มต้น และข้อมูล เพื่อให้มอนิเตอร์นำค่าข้อมูลไปเก็บในหน่วยความจำ
Y - Display Monitor Memory	-เมนูเป็นผู้รับคำสั่งจากผู้ใช้ในรูปแบบของเมนู ไม่ว่าจะเป็นคำสั่ง"B"หรือ"R"ที่เป็นการเรียกดูค่า เมนูจะส่ง"Y"มาให้กับมอนิเตอร์พร้อมด้วยแอดเดรสเพื่อให้มอนิเตอร์ส่งค่าในหน่วยความจำของมอนิเตอร์ซึ่งใช้เก็บค่า Breakpointและรีจีสเตอร์ไปให้เมนูจัดแสดงผลบนจอภาพ
Z - Enter Monitor Memory	-เมนูเป็นผู้รับคำสั่งจากผู้ใช้ในรูปแบบของเมนู ไม่ว่าจะเป็นคำสั่ง"B","C","R"หรือ "T" ที่มีการเขียนข้อมูลลงหน่วยความจำของมอนิเตอร์ เมนูจะส่ง "Z" มาให้กับมอนิเตอร์ พร้อมด้วยแอดเดรสและข้อมูล
G - GO	-เมนูเป็นผู้รับคำสั่งจากผู้ใช้ในรูปแบบของเมนู ไม่ว่าจะเป็นคำสั่ง"T"หรือ"G" เมนูจะส่ง"G" มาให้กับกับมอนิเตอร์ เพื่อให้ทำงานตามโปรแกรมผู้ใช้
I - In Port	-เมนูจะส่ง"I"และพอร์ตแอดเดรสมาให้กับมอนิเตอร์เหมือนในโหมดเทอร์มินัล
O - Out Port	-เมนูจะส่ง"O" พอร์ตแอดเดรส และข้อมูลมาให้มอนิเตอร์ เหมือนในโหมดเทอร์มินัล
X - Reset	-เมนูจะส่ง"X"มาให้มอนิเตอร์เพื่อรีเซตระบบ

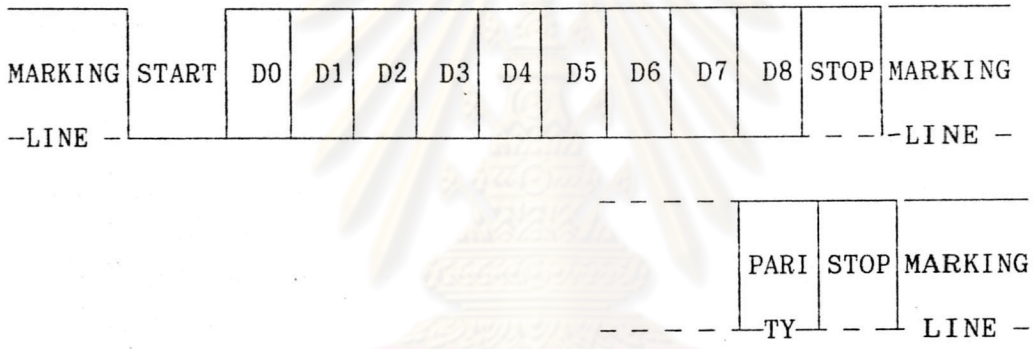
โปรแกรมมอนิเตอร์ที่ใช้ในการดีบักโปรแกรมของผู้ใช้ซึ่งอยู่บนบอร์ดอีมูเลชันประกอบด้วยโปรแกรมบนรอม 32 K และแรมอีก 8 K โดยอยู่คนละเพจแอดเดรสกับโปรแกรมของผู้ใช้ โปรแกรมมอนิเตอร์เขียนด้วยภาษา PL/M [22] และภาษาแอสเซมบลี้ซึ่งมีอยู่ 2 โปรแกรมคือ โปรแกรมมอนิเตอร์ของซีพียูเบอร์ 8085 และโปรแกรมมอนิเตอร์ของซีพียูเบอร์ Z-80 ทั้งนี้เนื่องจากโครงสร้างของซีพียูและรูปแบบของคำสั่งต่างกัน แต่โปรแกรมมอนิเตอร์ทั้งสองนี้จะมีโครงสร้างใกล้เคียงกัน ซึ่งสามารถแสดงขั้นตอนการทำงานของโปรแกรมมอนิเตอร์ได้ดังรูปที่ 5.4



รูปที่ 5.4 แสดงขั้นตอนการทำงานของโปรแกรมมอนิเตอร์

โปรแกรมมอนิเตอร์จะมีลักษณะของโปรแกรมหลัก ซึ่งมีส่วนที่กำหนดรูปแบบการติดต่อรับส่งข้อมูลระหว่างเครื่องไมโครคอมพิวเตอร์ IBM PC กับบอร์ดอีมูเลชัน ส่วนที่กำหนดค่าเริ่มต้นต่างๆ และส่วนที่รับคำสั่งมาจำแนกประเภท จากนั้นก็จะเรียกใช้โปรแกรมย่อยเพื่อให้ซีพียูทำงานตามคำสั่งนั้น ๆ หากมีข้อผิดพลาดก็จะแสดงข้อความ Error ออกมาที่จอภาพ

รูปแบบของการติดต่อรับส่งข้อมูลระหว่างบอร์ดอีมูเลชันกับเครื่องไมโครคอมพิวเตอร์ IBM PC หรือเทอร์มินัล ถูกกำหนดให้เป็นการรับส่งข้อมูลแบบอะซิงโครนัสอนุกรม (Asynchronous Serial Communication) ผ่านทางพอร์ต RS-232C ดังรูปที่ 5.5 ซึ่งกำหนดให้มีบิตเริ่มต้น (Start bit) 1 บิต บิตข้อมูล 8 บิต ไม่มีพาริตี (No Parity)และบิตท้าย (Stop bit) 1 บิต



รูปที่ 5.5 แสดงรูปแบบการรับส่งข้อมูลแบบอะซิงโครนัสของระบบพัฒนา

ส่วนขั้นตอนการทำงานของแต่ละคำสั่ง เราสามารถอธิบายดังต่อไปนี้

1. คำสั่งป้อนโปรแกรมเป็นภาษาแอสเซมบลีทีละบรรทัด "A" หลังจากรับคำสั่งจากคีย์บอร์ดแล้วตรวจได้ว่าเป็นคำสั่ง "A" ถูกต้อง โปรแกรมหลักจะเรียกโปรแกรมย่อยซึ่งมีลำดับการทำงานดังนี้

- แสดงแอดเดรสเริ่มต้นที่ต้องการป้อนโปรแกรม และรอรับโปรแกรมเป็นภาษาแอสเซมบลี้

- เมื่อผู้ใช้กด Enter โปรแกรมย่อยนี้จะนำคำสั่งที่เขียนเป็นนิวโมนิกนั้นไปเปิดตารางเทียบออพโคด (Opcode) แล้วแปลโปรแกรมออกมาในรูปของภาษาเครื่อง

- นำมาแสดงต่อท้ายบรรทัดเดิม และนำค่าที่ได้ไปเก็บในหน่วยความจำของผู้ใช้ก่อนขึ้นบรรทัดใหม่

- แสดงค่าแอดเดรสถัดจากโปรแกรมบรรทัดที่แล้ว และรอรับการป้อนโปรแกรมต่อไป

- รับการป้อนโปรแกรมทีละบรรทัดไปจนกว่าผู้ใช้จะกด Enter โดยที่ไม่ได้ป้อนโปรแกรมในบรรทัดนั้น แล้วจึงกลับไปที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ และหากมีการป้อนคำสั่งผิดรูปแบบก็ให้แสดง Error บนจอ

2. คำสั่งกำหนดจุดหยุดปฏิบัติการ "B" หลังจากรับคำสั่งถูกต้องแล้วโปรแกรมหลักจะเรียกโปรแกรมย่อยซึ่งมีลำดับการทำงานดังนี้

- นำค่าที่ต่อท้ายคำสั่งมาเก็บลงในหน่วยความจำมอนิเตอร์ เพื่อบอกโปรแกรมมอนิเตอร์ว่า มีการ Enable Breakpoint ที่แอดเดรสใด

- ถ้าไม่มีแอดเดรสต่อท้ายคำสั่งก็แสดงว่า เป็นการเรียกดูค่าแอดเดรสที่จะหยุดปฏิบัติการ โปรแกรมย่อยจะดูค่าในหน่วยความจำมอนิเตอร์ว่ามีการ Enable หรือไม่ หากไม่มีก็จะแสดงข้อความ CLEAR! บนจอ แต่ถ้ามีก็แสดงค่าแอดเดรสที่จะหยุดปฏิบัติการบนจอก่อนกลับไปที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดง ">" บนจอ

3. คำสั่งยกเลิกจุดหยุดปฏิบัติการ "C" หลังจากรับคำสั่งถูกต้องแล้วโปรแกรมหลักจะไปเรียกโปรแกรมย่อยให้เขียนค่าในหน่วยความจำมอนิเตอร์ เพื่อยกเลิกการ Enable Breakpoint แล้วกลับไปที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

4. คำสั่งอ่านค่าข้อมูลจากหน่วยความจำ "D" หลังจากรับคำสั่งถูกต้องโปรแกรมหลักจะเรียกโปรแกรมย่อยให้ทำการอ่านข้อมูลจากเพจของผู้ใช้ตามขบวนการที่กล่าวมาแล้วจากนั้นจะนำค่าที่ได้มาแสดงบนจอแล้วกลับไปที่โปรแกรมหลัก เพื่อขึ้นบรรทัดใหม่แสดงเครื่องหมาย ">" บนจอ

5. คำสั่งเขียนค่าข้อมูลลงหน่วยความจำ "E" หลังจากรับคำสั่งถูกต้องโปรแกรมหลักก็จะทำขบวนการเช่นเดียวกับที่กล่าวมาแล้วทีละไบต์โดยโปรแกรมย่อย ก่อนที่จะกลับมาทำงานที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

6. คำสั่งเขียนค่าในหน่วยความจำเป็นกลุ่ม "F" จะทำคล้ายกับการทำงานในคำสั่ง "E" แต่จะทำทีละไบต์จนครบ ก่อนที่จะกลับมาทำงานที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

7. คำสั่งรันโปรแกรมของผู้ใช้ "G" จะมีขบวนการทำงานดังที่กล่าวมาแล้วในหัวข้อก่อน หากมีการสั่งหยุดก็จะกลับมาที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

8. คำสั่งขอความช่วยเหลือ "H" โปรแกรมหลักจะเรียกโปรแกรมย่อยเพื่อแสดง

ข้อความที่บอกรูปแบบการใช้งานคำสั่งต่าง ๆ บนจอภาพ ก่อนกลับมาที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

9. คำสั่งอ่านพอร์ตไอ/โอ "I" หลังจากรับคำสั่งถูกต้องโปรแกรมหลักจะเรียกโปรแกรมย่อยให้อ่านค่าจากพอร์ตไอ/โอของผู้ใช้ตามแอดเดรสที่ต่อท้ายคำสั่งและแสดงค่าบนจอก่อนขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

10. คำสั่งเขียนพอร์ตไอ/โอ "O" หลังจากรับคำสั่งถูกต้องโปรแกรมหลักจะเรียกโปรแกรมย่อยให้เอาค่าท้ายสุดของคำสั่งไปออกที่พอร์ตไอ/โอ ก่อนขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

11. คำสั่งดู/แก้ไข ค่าในรีจีสเตอร์ "R" หลังจากรับคำสั่งถูกต้องโปรแกรมหลักจะเรียกโปรแกรมย่อยซึ่งมีลำดับการทำงานดังนี้

- ให้นำเอาค่าที่เก็บอยู่ในหน่วยความจำมอนิเตอร์ซึ่งเป็นค่ารีจีสเตอร์ของผู้ใช้มาแสดงบนจอ

- หากมีชื่อรีจีสเตอร์ต่อท้ายคำสั่งแสดงว่าต้องการแก้ไขค่าของรีจีสเตอร์ตัวนั้น ก็จะนำเอาค่าใหม่ไปเก็บในหน่วยความจำมอนิเตอร์ที่เดิมของรีจีสเตอร์ตัวนั้น

- กลับมาที่โปรแกรมหลักเพื่อจะขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

12. คำสั่งให้ทำงานทีละคำสั่ง "T" โปรแกรมมอนิเตอร์จะทำงานตามที่ได้กล่าวมาแล้วในหัวข้อก่อนคือ จะทำงานเหมือนคำสั่ง "B" รวมกับคำสั่ง "G"

13. คำสั่งแสดงข้อมูลในหน่วยความจำเป็นภาษาแอสเซมบลี "U" หลังจากรับคำสั่งถูกต้องโปรแกรมหลักจะเรียกโปรแกรมย่อยซึ่งมีลำดับการทำงานดังนี้

- ให้ทำการอ่านข้อมูลจากหน่วยความจำเพจผู้ใช้ในแอดเดรสที่กำหนด แล้วเทียบเป็นภาษาแอสเซมบลี้ด้วยการเปิดตาราง

- หากเป็นคำสั่งที่ใช้มากกว่า 1 ไบต์ก็จะอ่านข้อมูลในแอดเดรสถัดไปมาใหม่

- หลังจากแปลเป็นคำสั่งสมบูรณ์แล้วก็แสดงบนหน้าจอ

- ขึ้นบรรทัดใหม่เพื่อไปอ่านข้อมูลมาทำการแปลอีกจนกว่าจะถึงแอดเดรสที่กำหนด จึงกลับไปสู่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่ และแสดงเครื่องหมาย ">" บนจอ

14. คำสั่งรีเซต "X" โปรแกรมหลักจะไปเรียกโปรแกรมย่อยให้เขียนค่าแอดเดรส 0000 ลงสู่หน่วยความจำมอนิเตอร์ ณ ตำแหน่งที่เก็บค่า PC ของผู้ใช้ก่อนจะกลับมาที่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

15. คำสั่งโหลดไฟล์ "L" โปรแกรมหลักจะเรียกโปรแกรมย่อยให้ทำการรอรับข้อมูลในรูปแบบของ Intel Hex formatจากเครื่องไมโครคอมพิวเตอร์ IBM PC ทางพอร์ต

RS-232C แล้วทำการเขียนค่าที่ได้รับลงในหน่วยความจำผู้ใช้ที่แอดเดรสต่าง ๆ ก่อนจะกลับไปสู่โปรแกรมหลักเพื่อขึ้นบรรทัดใหม่และแสดงเครื่องหมาย ">" บนจอ

16. คำสั่งเก็บข้อมูลในหน่วยความจำเป็นไฟล์ "W" โปรแกรมหลักเรียกโปรแกรมย่อยให้ทำการอ่านข้อมูลจากหน่วยความจำผู้ใช้ที่แอดเดรสที่กำหนด ส่งไปยังเครื่องไมโครคอมพิวเตอร์ IBM PC ผ่านทางพอร์ต RS-232C ในรูปแบบของ Intel Hex Format ทีละไบต์จนครบกำหนดแล้วจึงกลับไปที่โปรแกรมหลัก เพื่อขึ้นบรรทัดใหม่และแสดง ">" บนจอ

17. คำสั่งอ่านข้อมูลจากหน่วยความจำผู้ใช้ในโหมดโฮสต์"J" จะมีการทำงานคล้ายกับคำสั่ง "W" ในโหมดเทอร์มินัล

18. คำสั่งเขียนข้อมูลลงหน่วยความจำผู้ใช้ในโหมดโฮสต์"K" จะมีการทำงานคล้ายกับคำสั่ง "L" ในโหมดเทอร์มินัล

19. คำสั่งอ่านข้อมูลในหน่วยความจำมอนิเตอร์ "Y" หลังจากโปรแกรมหลักรับคำสั่งเรียบร้อยแล้ว จะเรียกโปรแกรมย่อยเพื่ออ่านข้อมูลในหน่วยความจำมอนิเตอร์จากแอดเดรสที่กำหนดแล้วจัดข้อมูลเป็นรูปแบบของIntel Hex format ก่อนจึงส่งออกไปที่พอร์ตของ 8251 เพื่อไปยังเครื่องไมโครคอมพิวเตอร์ IBM PC จนครบตามที่เมนูต้องการ

20. คำสั่งเขียนข้อมูลในหน่วยความจำมอนิเตอร์ "Z" หลังจากรับคำสั่งแล้วโปรแกรมหลักจะเรียกโปรแกรมย่อยเพื่อรับข้อมูลจากเครื่องไมโครคอมพิวเตอร์ IBM PC ในรูปแบบของ Intel Hex format ทีละเรคคอร์ดแล้วนำค่าที่ได้เก็บลงหน่วยความจำมอนิเตอร์ก่อนที่จะรับเรคคอร์ดถัดไป จนกว่าจะครบตามที่เมนูต้องการ

5.4 การใช้งานบอร์ดอีมูเลชัน

การดีบักโปรแกรมด้วยบอร์ดอีมูเลชันในระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ดมีขั้นตอนดังนี้

1. ตรวจดูการต่อ Jumper การเลือกใช้ชุด PAL และรอมมอนิเตอร์บนบอร์ดอีมูเลชันให้ตรงกับเบอร์ซีพียูที่ใช้(8085 หรือ Z-80)

2. เลือกใช้อัตราเร็วในการรับส่งข้อมูลว่าจะเป็น 1200, 2400, 4800 หรือ 9,600 บิตต่อวินาทีด้วยการเลือกต่อ Jumper บนบอร์ดอีมูเลชัน โดยปกติควรตั้งไว้ที่ 9,600

3. ประกบบอร์ดอีมูเลชันกับด้านบนของบอร์ดซีพียูและหน่วยความจำให้ถูกต้อง

4. ต่อสายเคเบิล RS-232C ระหว่างขั้วต่อสายบนบอร์ดอีมูเลชันกับพอร์ตอนุกรมของเครื่องไมโครคอมพิวเตอร์ IBM PC หรือเทอร์มินัลซึ่งมีการเชื่อมต่อสายสัญญาณภายในดังนี้

	Emulation Board		IBM PC or Terminal
ขาที่ 1	GND	>----------<	Frame GND
ขาที่ 2	RxD	>----------<	TxD
ขาที่ 3	TxD	>----------<	RxD
ขาที่ 4	CTS	>----------<	RTS
ขาที่ 5	RTS	>----------<	CTS
ขาที่ 6	DTR	>----------<	DSR
ขาที่ 7	GND	>----------<	Signal GND
ขาที่ 20	DSR	>----------<	DTR

5. ถ้าต้องการต่อกับฮาร์ดแวร์ของระบบที่ต้องการจะพัฒนา ก็ให้เสียบบอร์ดต่าง ๆ ที่ได้ออกแบบเลือกใช้ไว้แล้วเข้าสู่บัสคอนเนกเตอร์บนบอร์ดแม่ แต่หากต้องการพัฒนาเฉพาะซอฟต์แวร์ก่อนก็สามารถเสียบบอร์ดซีพียูที่มีบอร์ดอีมูเลชันประกบอยู่ด้านบน เข้ากับคอนเนกเตอร์บนบอร์ดแม่เพียงช่องเดียวก่อน

6. ต่อไฟจากแหล่งจ่ายไฟตรง +5 V , ±12 V และกราวน์เข้ากับขั้วต่อทางด้านหลังของบอร์ดแม่

7. เปิดเครื่องไมโครคอมพิวเตอร์ IBM PC แล้วเรียกโปรแกรม PROCOMM เพื่อจำลองการทำงานเป็นเทอร์มินัล หรือถ้าใช้กับเทอร์มินัลจริงก็เปิดเครื่องแล้วตั้งค่าในการรับส่งข้อมูลดังนี้

- กำหนดการติดต่อกับพอร์ต COM1 หรือ COM2 ตามเบอร์พอร์ตของ IBM PC ที่ต่ออยู่กับบอร์ดอีมูเลชัน
- ตั้งอัตราเร็วในการรับส่งข้อมูลเป็น 1200, 2400, 4800 หรือ 9600 บิตต่อวินาที (bps) ให้ตรงกับการต่อ Jumper บนบอร์ดอีมูเลชัน (ตามข้อ 2 ในหน้าที่แล้ว)
- กำหนดรูปแบบของข้อมูลเป็น

  8 bit/character , 1 stop bit , no parity

8. เปิดไฟเข้าสู่ระบบพัฒนาไมโครโปรเซสเซอร์ระดับบอร์ด

9. หากการติดต่อผ่านพอร์ต RS-232C สำเร็จ จะปรากฏเมนูคำสั่ง และเครื่องหมาย ">" บนจอภาพ แต่หากการติดต่อสื่อสารไม่สำเร็จจะไม่มีอะไรปรากฏบนจอภาพให้ทำการตรวจสอบดังนี้

- ตรวจสอบสายเคเบิล RS-232C ว่าการต่อสายสัญญาณภายในตรงตามที่กำหนดหรือไม่

- ตรวจสอบสายเคเบิล RS-232C ว่าระหว่างปลายทั้งสองข้างต่อถึงกันหรือไม่

- ตรวจสอบแหล่งจ่ายไฟของระบบพัฒนาว่ามีแรงดัน +5 V และ $\pm$12 V เข้าสู่ระบบหรือไม่

- ตรวจสอบการตั้งค่าต่าง ๆ ในการรับส่งข้อมูลว่ามีการกำหนดพอร์ต COM ที่ใช้งาน อัตราเร็วในการรับส่งข้อมูล และรูปแบบของข้อมูลตรงตามบอร์ดอีมูเลชันหรือไม่

- ให้กดปุ่มรีเซตบนบอร์ดซีพียู

ถ้าทำทุกขั้นตอนข้างต้นแล้วไม่ประสบผลสำเร็จให้ตรวจสอบไอซีเบอร์ MC1488 และ MC1489 ก่อนแล้วจึงทำการตรวจสอบไอซีตัวอื่น ๆ ต่อไป

ชุดคำสั่งที่ช่วยในการดีบักของบอร์ดอีมูเลชันสามารถสรุปตามการใช้งานได้ดังต่อไปนี้

- คำสั่งแต่ละคำสั่งจะมี Syntax ทั่ว ๆ ไป คือ

  > Command [parameter] <CR>

  <CR> คือ Carriage Return

ส่วนรายละเอียดของแต่ละคำสั่งจะอธิบายต่อไป

คำสั่งในโหมดเทอร์มินัล ได้แก่

[A] Assemble one line Command

Syntax:

>A xxxx <CR>

xxxx คือ แอดเดรสเริ่มต้น ซึ่งเป็นเลขฐานสิบหก

Operation:

คำสั่งนี้ทำให้ผู้ใช้สามารถป้อนโปรแกรมภาษาแอสเซมบลีลงสู่หน่วยความจำทีละบรรทัดได้และจะทำการแปลเป็นภาษาเครื่องแสดงไว้ท้ายบรรทัดด้วย

Example:

- ตัวอย่างการป้อนโปรแกรมของ Z-80 โดยเริ่มจากแอดเดรส 1800H

```
>A 1800 <CR>
1800    LD B,0 <CR>          06 00
1802    LD C,0 <CR>          0E 00
1804    LD HL,1900 <CR>      21 00 19
1807    INC C <CR>           0C
1808    <CR>
>
```

[B] Breakpoint

Syntax:

>B [xxxx] <CR>

xxxx คือแอดเดรสของจุดหยุดปฏิบัติการ (ถ้าต้องการดูค่าเดิมก็ไม่ต้องใส่)

Operation:

คำสั่งนี้ใช้กำหนดจุดหยุดปฏิบัติการหากมีค่าแอดเดรสตามหลังคำสั่ง

หรือใช้ดูค่าแอดเดรสของจุดหยุดปฏิบัติการหากไม่มีค่าแอดเดรสตามหลัง

Example:

- กำหนดจุดหยุดปฏิบัติการที่แอดเดรส 1800H

  >B 1800 <CR>

  >

- ดูว่ามีการกำหนดจุดหยุดปฏิบัติการไว้ที่ใด

  >B <CR>

  Break Point Address 1800

  >

  หากไม่มีจะแสดงคำว่า "Break Point Clear!"

[C] Clear Breakpoint

Syntax:

>C <CR>

Operation:

คำสั่งนี้ใช้ยกเลิกจุดหยุดปฏิบัติการที่ได้ตั้งไว้แล้ว

Example:

> C <CR>

>

[D] Display User Memory or Dump Memory

Syntax:

>D xxxx [yyyy] <CR>

xxxx คือ แอดเดรสเริ่มต้น ที่จะดูค่าข้อมูล

yyyy คือ แอดเดรสท้ายสุด ถ้าดูแอดเดรสเดียวไม่ต้องใส่

Operation:

คำสั่งนี้ใช้ดูค่าข้อมูลในหน่วยความจำของผู้ใช้ซึ่งสามารถดูได้ทีละแอดเดรส หรือดูเป็นกลุ่ม

Example:

- ต้องการดูค่าข้อมูลที่แอดเดรส 1800H จะแสดงทีละครึ่งจอ (128 ไบต์)

```
>D 1800 <CR>
1800 06 00 06 00 21 00 19 0C-04 3C C3 00 18 00 00 FF
1810 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
1820 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
1830 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
1840 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
1850 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
1860 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
1870 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
>
```

- ต้องการดูค่าข้อมูลที่แอดเดรส 1800H ถึงแอดเดรส 1830H

```
>D 1800 1830 <CR>
1800 06 00 06 00 21 00 19 0C 04 3C C3 00 18 00 00 FF
1810 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1820 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1830 FF
>
```

[E] Edit User Memory

Syntax:

>E xxxx [nn]<CR>

xxxx คือ แอดเดรสที่ต้องการแก้ค่า

nn คือ ข้อมูลที่ต้องการใส่แทนค่าเดิม (ฐานสิบหก)

Operation:

คำสั่งนี้ใช้แก้ไขค่าข้อมูลในหน่วยความจำทีละไบต์ หากคำสั่งมีค่าของข้อมูลตามหลังแอดเดรส จะเป็นการแก้ไขข้อมูลในหน่วยความจำเพียงไบต์เดียว

Example:

- ต้องการเปลี่ยนค่าของข้อมูลที่แอดเดรส 1800H เป็น 16H

```
>E 1800 16 <CR>
>
```

- ต้องการแก้ไขค่าของข้อมูลเริ่มที่แอดเดรส 1800H

```
>E 1800 <CR>
1800 16 C3 <CR>
1801 00 00 <CR>
1802 00 18 <CR>
1803 00 <CR>
>
```

[F] Fill Block User Memory (with a data)

Syntax:

>F xxxx yyyy nn

xxxx คือ แอดเดรสเริ่มต้น

yyyy คือ แอดเดรสสุดท้าย

nn คือ ค่าของข้อมูลในฐานสิบหก

Operation:

คำสั่งนี้ใช้ป้อนค่าข้อมูลลงสู่หน่วยความจำ โดยเป็นค่าเดียวกันทั้งกลุ่ม

Example:

- ต้องการป้อนค่า 00H ที่แอดเดรส 1900H - 191FH

>F 1900 191F 00 <CR>

>

[G] GO

Syntax:

>G [xxxx] <CR>

xxxx คือ แอดเดรสที่จะให้เริ่มรัน

Operation:

คำสั่งให้ซีพียูเริ่มรันโปรแกรมที่แอดเดรสซึ่งผู้ใช้กำหนดมาให้ แต่ถ้าไม่มีก็คือสั่งให้ซีพียูเริ่มรันที่แอดเดรสซึ่งค่า PC ชี้อยู่

Example:

- สั่งรันโปรแกรมตั้งแต่แอดเดรส 1800H

>G 1800 <CR>

>

- สั่งรันโปรแกรมตั้งแต่แอดเดรสที่ PC ชี้อยู่

>G

>

[H] Help Menu

Syntax:

>H <CR>

Operation:

คำสั่งนี้จะแสดงเมนูคำสั่งของผู้ใช้บนจอ

Example:

- ผู้ใช้ต้องการดูการใช้งานคำสั่งต่าง ๆ

```
>H  <CR>
Assemble        A [address]
Breakpoint      B [address]
Clearbreak      C
Dump            D [address¦range]
Enter           E address [list]
Fill            F range list
Go              G [address]
Help            H
Load            L
Input           I address
Output          O address byte
Register        R [registername]
Trace           T
Unassemble      U [S] [address¦range]
Write           W range
reset           X
>
```

[I] In port

Syntax:

>I pp <cr>

pp คือ แอดเดรสพอร์ตไอ/โอ

Operation:

คำสั่งให้อ่านค่าจากพอร์ตไอ/โอตามแอดเดรสที่กำหนด

Example:

- ต้องการทราบค่าที่พอร์ตไอ/โอ แอดเดรส 0FH

>I 0F <CR>

00

>

[O] Out port

Syntax:

>O pp nn <CR>

pp คือ แอดเดรสพอร์ตไอ/โอ

nn คือ ค่าของข้อมูลในฐานสิบหก

Operation:

คำสั่งให้ส่งค่าข้อมูลที่กำหนด (nn) ออกไปที่พอร์ตไอ/โอแอดเดรสที่กำหนด

Example:

- ต้องการส่งค่า 55H ไปที่พอร์ตไอ/โอ แอดเดรส 0FH

>O 0F 55 <CR>

>

[R] Register Examined

Syntax:

>R [r] <CR>

r คือ ชื่อของรีจีสเตอร์ เช่น A,F,B,C,D,E,H,L,SP และ PC เป็นต้น

Operation:

หากคำสั่งนี้ไม่มีชื่อรีจีสเตอร์ต่อท้าย จะเป็นคำสั่งขอดูค่าในรีจีสเตอร์แต่ถ้ามีชื่อรีจีสเตอร์ต่อท้ายคำสั่งจะเป็นการแก้ไขค่าในรีจีสเตอร์โดยจะแสดงค่าเดิมให้ดูด้วย

Example:

- ต้องการดูค่าในรีจีสเตอร์ทั้งหมดของซีพียู 8085

```
>R <CR>
 PC   AF   BC   DE   HL   IM   SP
0000 0000 0000 0000 0000 0087 0000
>
```

- ต้องการตั้งค่ารีจีสเตอร์ PC ให้เป็น 1800H

```
>R PC <CR>
 1800 <CR>
>
```

[T] Trace or Single Step

Syntax:

>T <CR>

Operation:

คำสั่งให้ซีพียูทำงานทีละคำสั่งแล้วหยุดเพื่อให้ผู้ใช้สามารถตรวจดูค่าต่าง ๆ

Example:

- ต้องการทำ Single Step หลังจากตั้งค่า PC แล้ว

```
>T <CR>
>
```

[U] Unassemble

Syntax:

>U [xxxx yyyy] <CR>

xxxx คือ แอดเดรสเริ่มต้น

yyyy คือ แอดเดรสท้ายสุด

Operation:

คำสั่งให้แสดงข้อมูลในหน่วยความจำออกมาเป็นโปรแกรมภาษาแอสเซมบลี บนหน้าจอ หากไม่กำหนดแอดเดรสท้ายคำสั่งจะแสดงทีละ 10 บรรทัด

Example:

- ต้องการดูโปรแกรมที่ป้อนเข้าไปแล้วที่แอดเดรส 1800H-1807H ว่าถูกต้องหรือไม่

```
>U 1800  1804 <CR>
1800       LD B,00
1802       LD C,00
1804       LD HL,1900
>
```

[X] Reset

Syntax:

>X <CR>

Operation:

คำสั่งรีเซตให้ซีพียูกลับไปทำงานที่แอดเดรส 0000H

Example:

- ต้องการรีเซตระบบเพื่อกลับไปจุดเริ่มต้นโปรแกรมแอดเดรส 0000H

```
>X <CR>
>
```

[L] Load file

Syntax:

>L <CR>

Operation:

คำสั่งโหลดไฟล์ (Intel Hex Format) จากเครื่องไมโครคอมพิวเตอร์ IBM PC ไปยังหน่วยความจำบนบอร์ดอีมูเลชัน

Example:

- ต้องการโหลดไฟล์ LIFTO.HEX จากดิสค์ของเครื่องไมโครคอมพิวเตอร์ IBM PC ลงสู่หน่วยความจำเพื่อทดลองรัน และตรวจหาข้อผิดพลาดต่อไป โดยเครื่องไมโครคอมพิวเตอร์ IBM PC กำลังถูกจำลองการทำงานเป็นเทอร์มินัลด้วยโปรแกรม PROCOMM+

>L <CR>
Use load file to COM utility of your terminal

หลังจากปรากฏข้อความดังกล่าว ให้ใช้คำสั่ง Upload ของโปรแกรม PROCOMM+ โดยกดปุ่ม Page Up และเลือกการส่งแบบ ASCII แล้วใส่ชื่อไฟล์ LIFTO.HEX เมื่อโหลดไฟล์เรียบร้อยแล้วจะปรากฏคำว่า

Load ready.
>

[W] Write to file

Syntax:

>W xxxx yyyy <CR>

xxxx คือ แอดเดรสเริ่มต้น

yyyy คือ แอดเดรสท้ายสุด

Operation:

คำสั่งเก็บข้อมูลจากหน่วยความจำในช่วงแอดเดรสที่กำหนดลงสู่ดิสค์ของเครื่องไมโครคอมพิวเตอร์ IBM PCโดยเก็บเป็นไฟล์แบบ Intel Hex

Example:

- ต้องการเก็บโปรแกรมที่พัฒนาแล้วซึ่งอยู่ที่แอดเดรส 1800H-18FFH ในไฟล์ชื่อ LIFT.HEX

>W 1800 18FF <CR>

Use write COM to file utility of your terminal

Strike a key when ready ..

หลังจากปรากฏข้อความดังกล่าว ให้ใช้คำสั่ง Download ของโปรแกรม PROCOMM+ โดยกดปุ่ม Page Down และเลือกการส่งแบบ ASCII พร้อมกับตั้งชื่อไฟล์ตามต้องการก่อนกดแป้นพิมพ์อีก 1 ครั้ง เมื่อเก็บโปรแกรมเรียบร้อยแล้วจะปรากฏคำว่า

Write ready.

>

รายละเอียดของคำสั่งในโหมดโฮสต์

[J] Display User Memory [Y] Display Monitor Memory

Syntax:

J xxxx bb

or Y xxxx bb

xxxx คือ แอดเดรสเริ่มต้น

bb คือ จำนวนไบต์

Operation:

คำสั่งที่ใช้โดยเมนูบนเครื่องไมโครคอมพิวเตอร์ IBM PC ซึ่งต้องการอ่านข้อมูลดิบจากหน่วยความจำของระบบพัฒนา ในแบบของ Intel Hex format เพื่อจะนำไปทำการประมวลผลโดยโปรแกรมเมนูบนเครื่องไมโครคอมพิวเตอร์ IBM PC ต่อไป

[K] Enter User Memory [Z] Enter Monitor Memory

Syntax:

K or Z

Operation:

คำสั่งที่ใช้โดยเมนูบนเครื่องไมโครคอมพิวเตอร์ IBM PC ซึ่งต้องการป้อนข้อมูลลงสู่หน่วยความจำของระบบพัฒนา โดยจะส่งข้อมูลมาในรูปแบบของ Intel Hex format