คีย์รหัสสำหรับระบบรักษาความปลอดภัย

KEY CODE FOR SECURITY SYSTEM

โดย

พรชัย ทองเจิม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีอิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

หัวข้อปริญญานิพนธ์ : คีย์รหัสสำหรับระบบรักษาความปลอดภัย

KEY CODE FOR SECURITY SYSTEM

โดย : นาย พรชัย ทองเจิม เลขประจำตัว 33-131211

อาจารย์ที่ปรึกษา : ดร.กนก เจนจิระพงศ์เวช

ภาควิชา : เทคนิคอุตสาหกรรม

ปีการศึกษา : 2538

คณะวิศวกรรมศาตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์นี้ เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการตรวจสอบ ปริญญานิพนธ์

..... ประธานกรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

วันที่.....เดือน.....พ.ศ.....

ลิขสิทธิ์ของ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

คีย์รหัสสำหรับระบบรักษาความปลอดภัย

KEY CODE FOR SECURITY SYSTEM

โดย นายพรชัย ทองเจิม

อาจารย์ที่ปรึกษา ดร.กนก เจนจิระพงศ์เวช

บทคัดย่อ

ปริญญานิพนธ์นี้ เป็นการศึกษาถึงการทำงานของไมโครโปรเซสเซอร์ เบอร์ Z-80 และ การเชื่อต่อเข้ากับอุปกรณ์ภายนอกเช่น หน่วยความจำ พอร์ต อินพุท เอาท์พุท เบอร์ 8255 เพื่อมาควบคุม KEY BORD,LCD DISPLAY นอกจากนี้ยังศึกษาถึง การเขียนโปรแกรมภาษา ASSAMBLY เพื่องานควบคุมโดยผู้ทดลอง ได้ทำ SIGLE BORD ขึ้นมาควบคุม KEY BORD, มอนิเตอร์ แบบ LCD DISPLAY และ KEY LOCK ได้ผลดีพอ ที่จะสามารถนำไปใช้งานจริงได้ คิดว่าโครงงานนี้คงเป็นประโยชน์ต่อผู้ที่ต้องการออกแบบ งานควบคุมด้วยไมโครโปรเซสเซอร์ ในโอกาสต่อไป

ABSTRACT

This project introduce the microprocessor Z-80 and its operation by interfacing with memory input/output port of the IC chip 8255 for controlling the keybord and LCD display. Further more programming used hue is the assembly language. Herein,the single board is utilized to control the keybord, LCD display and keylock. This project will be an asset to designer in microprocessor controlled by various aspected.

สารบัญ

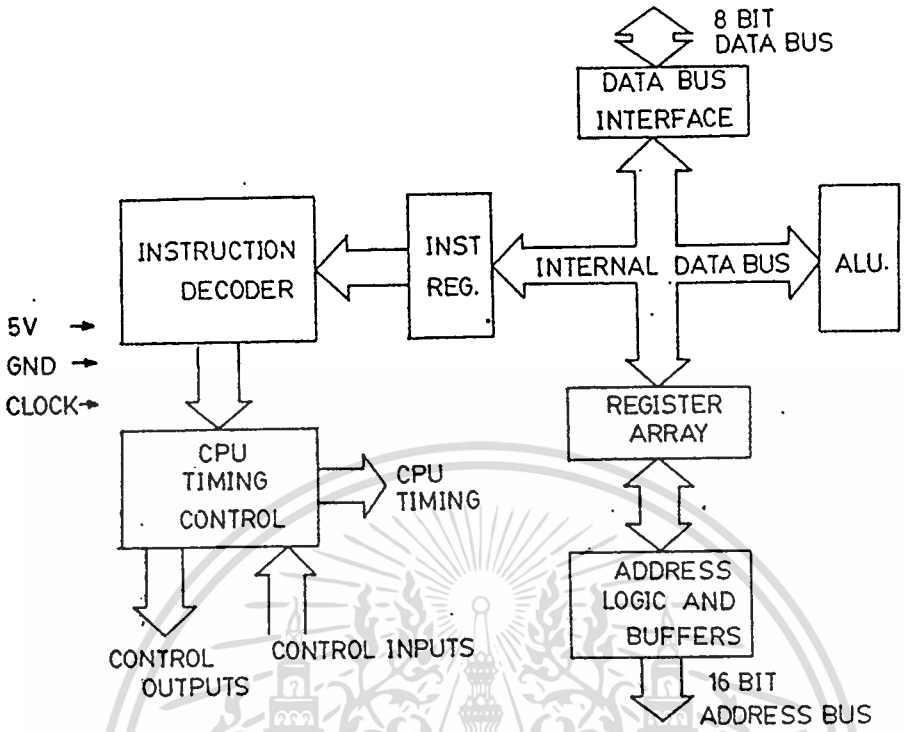
บทที่ 1

ไมโครโปรเซสเซอร์ (Z-80)

ไมโครโปรเซสเซอร์เบอร์ Z-80 เป็นไมโครโปรเซสเซอร์ขนาด 8 บิต ที่ได้พัฒนามาจากไมโครโปรเซสเซอร์ เบอร์ 8080 ของบริษัทอินเทล โดยทำ การแก้ไขข้อบกพร่องบางอย่างของไมโครโปรเซสเซอร์เบอร์ 8080 เช่นทำให้มีคำสั่งมากขึ้น มีวิธีอ้างอิงถึงตำแหน่งข้อมูล (AddressingMode) ใหม่และมีระบบฮาร์ดแวร์ที่มีความสามารถ และมีความสะดวกในการใช้งานมากขึ้น นอกจากนั้น ซีพียู Z-80 ยังสามารถใช้ซอฟแวร์ที่ใช้กับ ซีพียู 8080 ได้อีกด้วยเหตุผลในการศึกษาไมโครโปรเซสเซอร์โดยใช้ซีพียู Z-80 เนื่องจากโครงสร้างของซีพียู Z-80 เป็นที่เข้าใจง่ายทั้ง ด้านฮาร์ดแวร์ และการนำไปเชื่อมกับอุปกรณ์ภายนอกรวมทั้งชุดคำสั่งต่างๆ และ ระบบซอฟแวร์ ยังเข้าใจได้ง่าย และขีดความสามารถทำงานสูง

โครงสร้างซีพียู Z-80

โครงสร้างซีพียู Z-80 แสดงดังรูปที่ 1 จากรูปเห็นได้ว่ามีซีพียูบัสอยู่ 3 ชนิด คือ บัสตำแหน่ง บัสข้อมูล และบัสควบคุม ซึ่งบัสเหล่านี้ใช้สำหรับทำการเชื่อมต่ออุปกรณ์ภายนอกและภายในไมโครโปรเซสเซอร์จะประกอบด้วยพื้นฐานต่างๆ คือ ALU แอคคิวมูเลเตอร์ รีจีสเตอร์ตำแหน่งและแฟล็ก นอกจากนี้ยังมีรีจีสเตอร์ใช้งานต่าง ๆ อีกคือ B,C,D,E,H,L,B',C',D',E',H',และ L' ส่วนรีจีสเตอร์ตำแหน่งคือ PC,SP,IX,IY ซึ่งรีจีสเตอร์ต่าง ๆ เหล่านี้จะได้กล่าวถึงรายละเอียดต่อไป ส่วนทางด้านซ้ายสุดคือส่วนของหน่วยควบคุม ซึ่งส่วนนี้มีหน้าที่ถอดรหัสคำสั่งแล้วส่งสัญญาณควบคุมไปยังส่วนต่าง ๆ ทั้งภายในและภายนอก ซีพียู รวมทั้งรับสัญญาณควบคุมภายนอกเข้ามาด้วยสัญญาณควบคุมซีพียู Z-80 ภายนอกมี 13 สัญญาณ ซึ่งอาจแบ่งได้ 2 อย่าง คือ สัญญาณควบคุม ซีพียู (CPU and system control) ซึ่งสัญญาณต่าง ๆ จะทำให้มีผลต่อซีพียูและมีผลต่อระบบไมโครคอมพิวเตอร์ด้วย บัสข้อมูลเป็นบัสที่มีขนาด 8 บิต ที่ใช้เป็นทางเดินของข้อมูลระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำ หรืออุปกรณ์อินพุท/เอาท์พุทต่าง ๆ บัสตำแหน่งเป็นบัสที่มีขนาด 16 บิตเพื่อใช้ในการอ้างถึงหน่วยความจำ ดังนั้นจะทำให้การอ้างอิงหน่วยความจำสามารถทำได้ถึง 2^{16} ตำแหน่ง (65536 ตำแหน่ง หรือ 64 K) คือตำแหน่งที่ 0-65535 บล็อกที่ มีเครื่องหมาย +/- ที่อยู่ด้านล่างซ้ายของรีจีสเตอร์ หมายถึง การเพิ่มหรือลดข้อมูล (Increment Decrement) ที่มีอยู่ในรีจีสเตอร์ตำแหน่ง หรือของรีจีสเตอร์ๆ คือ SP,PC,BC,DE,HL ซึ่งรีจีสเตอร์ต่างๆ เป็นรีจีสเตอร์ในการกำหนดตำแหน่งโดยตรง (pure address register) ของการอ้างถึงตำแหน่งหน่วยความจำแบบตรง (direct address mode) ซีพียู Z-80 มีคำที่เกี่ยวกับอินพุท/เอาท์พุท



รูปที่ 1 โครงสร้างภายในของซีพียู Z-80

โดยเฉพาะไม่ใช้ลักษณะ อินพุท/เอาท์พุท Memary map ped (Memarymap I/O ใช้เป็นส่วนหนึ่งของหน่วยความจำเพื่อเป็นตำแหน่งของอุปกรณ์อินพุท/เอาท์พุท) ในการให้ไมโครโปรเซสเซอร์ทำงานตามที่ต้องการ ทำได้โดยการเขียนโปรแกรมไว้ในหน่วยความจำ จากนั้นให้ไมโครโปรเซสเซอร์ ออกคำสั่ง จากหน่วยความจำเพื่อ มาปฏิบัติคำสั่งต่าง ๆ นั้นเราไม่จำเป็นเข้าถึงส่วนรายละเอียดต่างๆ ของไมโครโปรเซสเซอร์ทั้งหมด โดยเราสนใจเฉพาะรีจีสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการเขียนโปรแกรมเท่านั้น รีจีสเตอร์ภายในที่สามารถอ่านเขียนได้ถึง 208 บิตโดยแยกเป็น รีจีสเตอร์ขนาด 8 บิต 18 รีจีสเตอร์ และ รีจีสเตอร์ขนาด 16 บิต อีก 4 รีจีสเตอร์ รีจีสเตอร์ต่างๆ ภายใน Z-80 เป็นลักษณะของสแตติกแรม และรีจีสเตอร์เหล่านี้แบ่งออกเป็น 3 ประเภท คือ

1. รีจีสเตอร์ใช้งานทั่วไป (General purpose register)
2. แอคคิวมูเลเตอร์ และ รีจีสเตอร์สถานะ (Accumulator and Flag register)
3. รีจีสเตอร์ใช้งานเฉพาะอย่าง (Special purpose register)

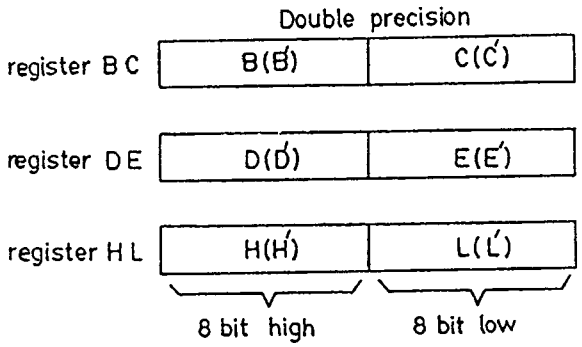
A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

I	R
INDEX REG. IX	
INDEX REG. IY	
STACPOINTER. SP	
PROGRAM COUTER. PC	

รูปที่ 2 รีจีสเตอร์ของซีพียู Z-80

<u>รีจีสเตอร์ใช้งานทั่วไป</u> (General purpose register) รีจีสเตอร์ ใช้งานทั่วไปภายในซีพียู Z-80 มี 2 กลุ่มที่เหมือนกัน แต่ละกลุ่มประกอบด้วยรีจีสเตอร์ขนาด 8 บิตจำนวน 6 รีจีสเตอร์ กลุ่มหนึ่งคือกลุ่มรีจีสเตอร์หลัก (Main register set) ซึ่งประกอบด้วย B,C,D,E,H,L ประโยชน์ของรีจีสเตอร์2 ชุดนี้คือชุดของรีจีสเตอร์สำรอง (Alternate register set) ซึ่งประกอบด้วย B',C',D',E',H',L' ประโยชน์ของการมีรีจีสเตอร์ 2 ชุดนี้คือ เพื่อให้ผู้เขียนโปรแกรมสามารถสลับระหว่างข้อมูล ในชุดของรีจีสเตอร์ทั้งสองได้อย่างรวดเร็ว ทั้งนี้เพื่อความเร็ว ในการทำงานของโปรแกรมเนื่องจากเข้าถึงข้อมูลของโปรแกรมภายในรีจีสเตอร์ของซีพียู มีความรวดเร็วกว่า การเข้าถึงข้อมูลในหน่วยความจำภายนอก นอกจากนี้รีจีสเตอร์สำรองอาจใช้เพื่อเก็บข้อมูลในรีจีสเตอร์หลักเมื่อ ซีพียูอินเตอร์รัพต์ภายนอก รีจีสเตอร์ใช้งานทั่วไปนี้อาจเป็นที่เก็บข้อมูลขนาด 8 บิต แบบธรรมดา หรือเก็บข้อมูลขนาด 16 บิต โดยการจับคู่ (Pair) ระหว่างรีจีสเตอร์ก็ได้ ซึ่งคู่รีจีสเตอร์ของรีจีสเตอร์หลักคือ BC,DE,HL ส่วนรีจีสเตอร์อีกชุดหนึ่งคือ BC', DE' และ HL' ประโยชน์ของคู่รีจีสเตอร์ เช่น ใช้เป็นตัวกำหนดตำแหน่ง (Pointer) ของข้อมูลในหน่วยความจำ หรือใช้เพื่อกระทำทางคณิตศาสตร์แบบ 16 บิต ที่เรียกว่า Double-precision arithmetic

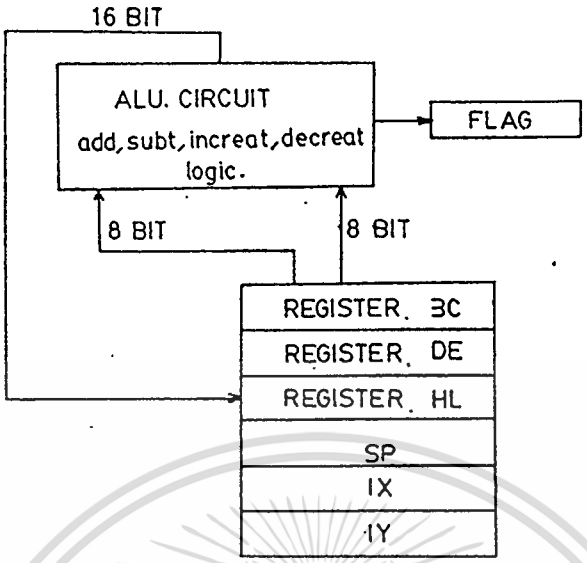
การกระทำทางคณิตศาสตร์แบบ Double-precision หมายถึงการบวก การลบ เพิ่มค่าขึ้น 1 (Increment) การลดค่าลง 1 (Decrement)ของข้อมูลขนาด 16 บิตที่อยู่ในคู่รีจีสเตอร์ หรืออยู่ในรีจีสเตอร์แบบ 16 บิต ซึ่งการกระทำทางคณิตศาสตร์หรือลอจิก ส่วนมากของ



รูปที่ 3 การจับคู่รีจีสเตอร์

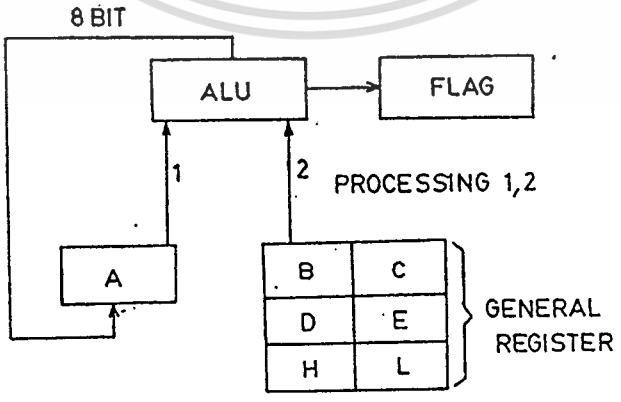
ซีพียู Z-80 เป็นกระทำของข้อมูลขนาด 8 บิต ซึ่งเรียกว่า Single-precision แต่การกระทำทางคณิตศาสตร์ขนาด 16 บิต ของซพียู Z-80 นี้สามารถเปลี่ยนแปลงตัวชี้ (Pointer) ได้สะดวกโดยการเปลี่ยนแปลงข้อมูล ภายในคู่รีจีสเตอร์ นอกจากนี้ยังเห็นได้ว่าถ้าไม่มีการบวกลบเลขแบบ 16 บิต จะต้องทำการบวกลบแบบ 8 บิตถึง 2 ครั้งซึ่งทำให้การเขียนโปรแกรมยุ่งยากขึ้น รูปที่ 3 แสดงถึงการจับคู่รีจีสเตอร์ เพื่อการกระทำแบบ Double-precision

<u>แอคคิวมูเลเตอร์และรีจีสเตอร์สถานะ</u> (Accumulator and Flag reg.)จากโครงสร้างของซีพียูจะเห็นได้ว่า โปรเซสเซอร์ทั่วไปจะต้องมี แอคคิวมูเลเตอร์ เพื่อเก็บข้อมูลตัวกระทำหลัก (Main operand) ตัวหนึ่ง เพื่อที่จะกระทำทางคณิตศาสตร์ หรือลอจิกกับข้อมูลที่มาจากรีจีสเตอร์ต่าง ๆ หรือข้อมูลที่มาจากหน่วยความจำภายนอก และผลจากการกระทำนั้นก็จะถูกนำมาเก็บไว้ใน แอคคิวมูเลเตอร์ ส่วนรีจีสเตอร์สถานะหรือแฟลกใช้เพื่อแสดงสถานะของแอคคิวมูเลเตอร์หรือรีจีสเตอรีที่เกี่ยวข้องการกระทำแสดงได้ดังรูปที่ 5 ไมโครโปรเซสเซอร์ Z-80 มีรีจีสเตอร์ที่ทำหน้าที่ แอคคิวมูเลเตอร์อยู่ 2 ตัวและแฟลก 2 ตัว แต่มีอย่างละตัวเท่านั้นที่สามารถทำงานกับหน่วยคณิตศาสตร์ และลอจิกได้คือ แอคคิวมูเลเตอร์และแฟลกที่อยู่ในส่วนของรีจีสเตอร์หลักซึ่งเรียกว่า A และ F ส่วนรีจีสเตอร์ A'และ F' ที่อยู่ในรีจีสเตอร์สำรอง ใช้เป็นที่เก็บข้อมูลชั่วคราวของแอคคิวมูเลเตอร์ A และ F เมื่อถึงคราวจำเป็นเท่านั้น ซึ่งคู่ของแอคคิวมูเลเตอร์และแฟลกทั้งสองชุดนี้ คือ AF และ AF' สามารถสลับข้อมูลในรีจีสเตอร์ทั้งคู่ ได้โดยคำสั่งในกลุ่มแลกเปลี่ยนข้อมูล (Exchange group) เหมือนการแลกเปลี่ยนข้อมูล ระหว่างรีจีสเตอร์หลักและรีจีสเตอร์สำรอง ของกลุ่มรีจีสเตอร์ทั่วไป รีจีสเตอร์สถานะหรือแฟลกรีจีสเตอร์ขนาด 8 บิต ซึ่งแต่ละบิตใช้แสดงสถานะซีพียู ที่จะเกิดหลังกระทำคำสั่ง ทางคณิตศาสตร์ หากคำสั่งทางลอจิกหรือกระทำคำสั่ง

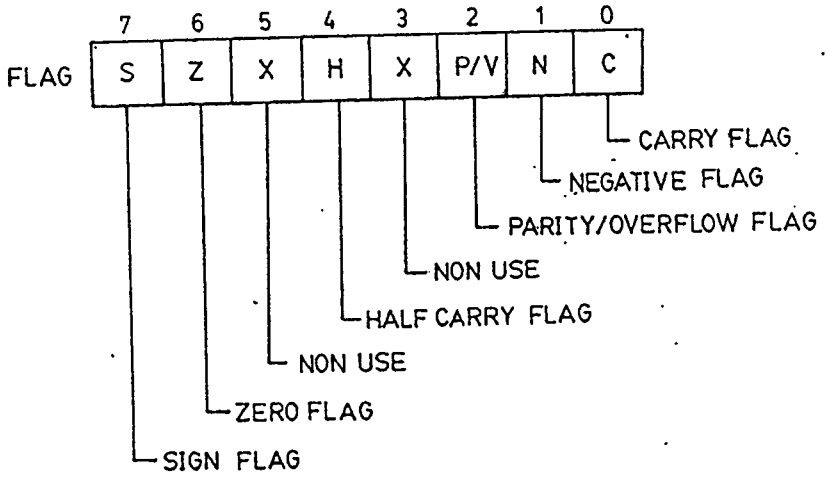


รูปที่ 4 การกระทำทางคณิตศาสตร์แบบ Double precision

อื่น ๆ ของซีพียูบิตต่าง ๆ ของแฟลกใช้แสดงสถานะที่แตกต่างกัน ซึ่งรูปแบบของรีจีสเตอร์นี้แสดงดัง รูปที่ 6 ของซีพียู กลุ่มคำสั่งที่มีผลต่อแฟลก เช่น คำสั่งในกลุ่มการกระทำคณิตศาสตร์ และ ลอจิก กลุ่มของ การเลื่อนหมุนข้อมูล กลุ่มของการทดสอบบิต เป็นต้น และในการทำงานข้าม (Jump or Branch) แบบมีเงื่อนไขที่ต้องหรือต้องการหรือไม่ รายละเอียดของแฟลกต่าง ๆ จะอธิบายได้ดังต่อไปนี้.



รูปที่ 5 การกระทำทางคณิตศาสตร์และลอจิก



รูปที่ 6 รูปแบบของแฟล็ก

<u>แฟลกตัวทด</u> (Carry flag)

แฟลกตัวทด หรือ แฟลก C แฟลกนี้จะเซตหรือรีเซต ขึ้นอยู่กับการทำงานของคำสั่งกลุ่มต่าง ๆ เช่นกลุ่มคณิตศาสตร์ กลุ่มของการเลื่อนและหมุนข้อมูลและตัวอย่าง เช่นการบวกเลขฐานสอง 8 บิต สองจำนวน หรือการลบเลข 8 บิต สองจำนวนดังรูปที่ 7

```
  0 1 1 0 1 0 1 1          0 1 0 0 1 0 1 0
+ 1 1 0 1 1 1 1 0        + 0 0 1 1 1 1 0 1
[1] 0 1 0 0 1 0 0 1      [0] 1 0 0 0 0 1 1 1
        (A)                      (B)

  0 1 1 0 1 0 1 1          0 1 0 1 1 0 1 0
- 0 0 1 0 1 0 0 1        - 1 0 0 0 1 1 0 0
[0] 0 1 0 0 0 0 1 0      [1] 1 1 0 0 0 1 1 0
        (C)                      (D)
```

รูปที่ 7 (A),(B) - ผลของแฟลก C หลังจากคำสั่งบวก
(C),(D) - ผลของแฟลก C หลังจากคำสั่งลบ

จากรูปที่ 7 (A) เห็นได้ว่าผลของการได้ผลรับเกิน 8 บิต ดังนั้นบิตที่เกินมาเป็น 9 บิต ที่จะทำให้ C เซ็ตเป็น 1 ส่วนรูปที่ 7(B) ผลของการบวกไม่เกิน 8 บิต แฟลก

C รีเซตเป็น 0 ในทำนองเดียวกันเมื่อทำคำสั่งลบแฟลก C เป็นแฟลกที่แสดงการขอยืม (borrow) ดังรูปที่ 7 (D) เมื่อมีการลบมีการขอยืมแฟลก C นี้ก็จะเป็นรีเซตเป็น 0 แต่ถ้า ซีพียูทำคำสั่งกลุ่มลอจิก ไม่ว่าคำสั่งใดก็ตาม แฟลก C นี้จะรีเซตเป็น 0 เสมอ

<u>แฟลกการลบ</u> (Negative flag)

แฟลกการลบ จะเป็นเซตหรือรีเซตขึ้นอยู่กับคำสั่งที่ใช้เกี่ยวข้องกับคณิตศาสตร์ นั้นคือ เมื่อทำคำสั่งประเภทการลบ แฟลกนี้จะเซต เช่นเมื่อ ซีพียู ทำคำสั่ง SUP CP DEC เป็นต้น แต่ถ้าซีพียูทำคำสั่งประเภทการบวก เช่น ADD INC แฟลกนี้จะรีเซต แฟลกฟารลบนี้มีประโยชน์มากเมื่อทำคำสั่ง DAA จะทำให้ ซีพียูรู้ว่าหน้านี้เป็นคำสั่งประเภทการบวก หรือการลบ เพื่อจะได้ทำการบวกหรือลบด้วย 6 เพื่อให้ได้เลข BCD ที่ต้องการ

<u>แฟลกพาริตี้/ค่าเกิน</u> (Parity/Overflow flag)

เป็นแฟลกที่อยู่บิต 2 ของแฟลกรีจีสเตอร์ แฟลกนี้มีประโยชน์สองอย่างคือ เมื่อ ซีพียูทำคำสั่งลอจิก แฟลกนี้เป็นแฟลกพาริตี้ แต่ถ้าซีพียูทำคำสั่งคณิตศาสตร์แฟลกนี้จะเป็นค่าเกิน ในกรณีของแฟลกพาริตี้ บิตนี้จะเซตเป็น 1 ทางผลของการกระทำทางลอจิก แล้วทำให้จำนวน 1 ของแอคคิวมูเลเตอร์ เป็นจำนวนคู่ (Even parity) และรีเซตเป็น 0 ถ้าจำนวน 1 ในแอคคิวมูเลเตอร์เป็นจำนวนคี่ (Odd parity) ดังตัวอย่างรูป 8

ในกรณีทำงานเป็นแฟลกค่าเกิน (Overflow หรือ v) คือเมื่อ ซีพียูทำคำสั่งทางคณิตศาสตร์ แฟลก V นี้จะเซตเมื่อผลลัพธ์จากการกระทำที่มีค่าเกิน และจะรีเซตผลการกระทำที่มีค่าไม่เกิน ค่าเกินนี้พิจารณาได้จากผลลัพธ์ที่จะได้ ที่แอคคิวมูเลเตอร์ เนื่องจากตัวเลขที่ใช้ในโครโปรเซสเซอร์เป็น 2's Complement Sign Binary นั่นคือ บิตสูงสุดเป็นบิตเครื่องหมาย และที่เหลือเป็นขนาดของตัวเลข ดังนั้นถ้าแอคคิวมูเลเตอร์มีขนาด 8 บิต มักแสดงเลขฐานสองได้ระหว่าง 0111 111 (+127) ถึง 1000 0000 (-128) ดังนั้น ถ้าผลการกระทำทางคณิตศาสตร์สูงกว่า +127 หรือต่ำกว่า -128 จะทำให้แฟลก V เซต แต่ถ้าผลการกระทำอยู่ระหว่าง +127 ถึง-128 จะรีเซต ดังตัวอย่างดัง รูปที่ 9

```
          10110110
     AND  10000101
ผลลัพธ์   10000100    --> "1"  มี 2 บิต เป็นจำนวนคู่ : แฟลก P=1


          01101001
```

OR 00110001

ผลลัพธ์ 01111001 --› "1" มี 5 บิต เป็นจำนวนคี่ : แฟลก P=0

รูปที่ 8 แม่อแฟลก P/V ทำหน้าที่เป็น พาริตี้แฟลก

+127	01111111	
+64	+01000000	(A)
+91	V[1] 10111111	

-125	10000011	
-126	+ 10000010	(B)
-251	V[1] 00000101	

+32	00100000	
+32	+ 00100000	(C)
+64	V[0] 01000000	

รูปที่ 9 แสดงการทำงานของแฟลก V

รูปที่ 9 (A)และ(B) แสดงให้เห็นว่าแฟลก V เซตสังเกตุได้ว่าในกรณีที่แฟลก V เซตเป็นไปได้ดังนี้คือ ถ้าบิตเครื่องหมายทั้งตัวตั้งและตัวบวกเป็น "0" (+) แล้วผลลัพธ์มีบิตเครื่องหมายเป็น "1"(-) แสดงว่าเกิดค่าเกิน (รูป 9 (A))ในทำนองเดียวกันถ้าบิตเครื่องหมายของตัวตั้งและตัวบวกเป็น "1"(-) ทั้งคู่แล้วผลลัพธ์มีบิตเครื่องหมายเป็น "0"(+) แสดงว่าเกิดค่าเกิน นอกเหนือจากนี้ แฟลก V จะรีเซต

<u>แฟลกตัวทดช่วย</u> (Half carry flag)

แฟลกตัวทดช่วย มีประโยชน์ในการกระทำทางคณิตศาสตร์ของตัวเลข BCD โดยที่แฟลกนี้จะมีการเซต เมื่อมีการทดหรือยืม ของการกระทำตัวเลข 4 บิต ทางด้านต่ำ (4 Least Sinificant Bit) ประโยชน์ของแฟลกนี้ ใช้เพื่อการปรับผลลัพธ์ของตัวเลขที่เกิดจากการกระทำทางคณิตศาสตร์ของเลข BCD ตัวอย่างของการเซตแฟลกตัวทดนี้ แสดงได้ดังรูปที่ 10

```
        0001 0001
      + 0010 0010      (A)  H=0
ผลลัพธ์ 001100011

        0100 1001
      + 0010 1000      (B)  H=1
ผลลัพธ์ 0111 0001
```

รูปที่ 10

แฟลกช่วยทด และแฟลกการลบ เป็นแฟลกเพื่อใช้ประโยชน์ส่วนใหญ่ในการกระทำทางคณิตศาสตร์ของเลข BCD

<u>แฟลกศูนย์</u> (Zero flag)

แฟลกศูนย์ ถ้าผลของการปฏิบัติคำสั่งแล้วทำให้ผลในรีจีสเตอร์ที่อ้างถึงมีค่าเป็น 0 และรีเซตถ้าผลลัพธ์ที่อ้างถึงไม่เป็น 0 ดังตัวอย่างรูปที่ 11

```
    01101001
AND 00000000
ผลลัพธ์ 00000000    แฟลก Z=1
```

รูปที่ 11 แสดงการทำงานของแฟลก Z

แฟลกศูนย์ นี้มีประโยชน์มากในการทำงานเคลื่อนแบบมีเงื่อนไขโดยใช้คำสั่งต่าง ๆ เช่น JP Z,nn JR Z,e JR NZ,nn เป็นต้น แฟลกศูนย์โดยทั่วไปใช้ในประโยชน์ดังนี้ เช่น

- ใช้สำหรับเปรียบเทียบเลขสองจำนวน
- ใช้ทดสอบบิต
- ใช้ตรวจสอบการลดค่าตัวนับ

<u>แฟลกเครื่องหมาย</u> (Sign flag)

แฟลกนี้ใช้ในการตรวจสอบบิตเครื่องหมาย ซึ่งเครื่องหมายบวกหรือลบนี้จะแสดงโดย

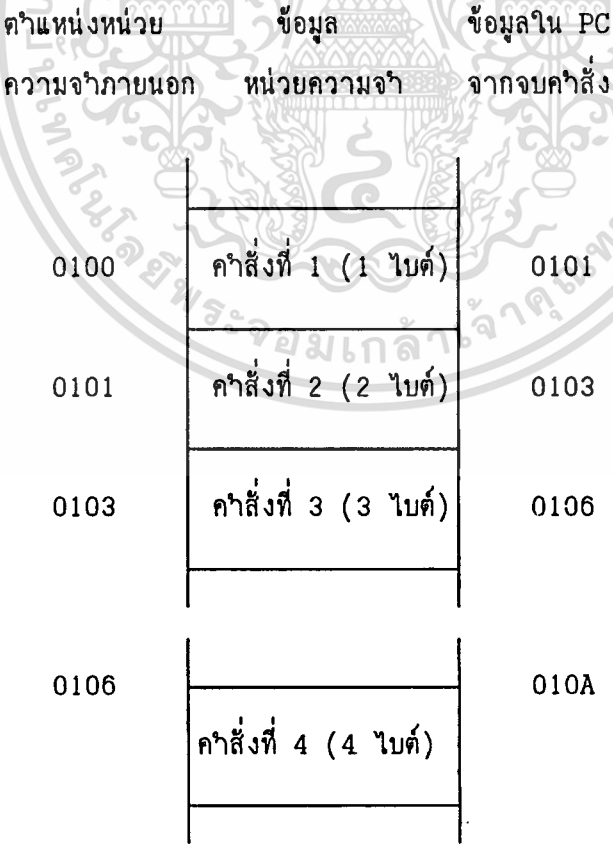
ตัวเลขบิตที่ 7 ของรีจีสเตอร์แฟลก ถ้าบิต 7 เท่ากับศูนย์แสดงว่าเป็นค่าบวก ถ้าบิต 7 เท่ากับ 1 แสดงว่าเป็นลบ แฟลกเครื่องหมายจะเซตถ้าหลังจากการปฏิบัติคำสั่งให้ผลลัพธ์ ที่มีเครื่องหมายเป็นลบ และรีเซตเมื่อผลลัพธ์ได้เครื่องหมายเป็นบวก

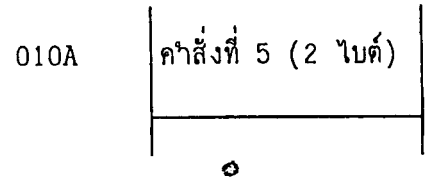
<u>รีจีสเตอร์ใช้งานเฉพาะอย่าง</u> (Special purposee register)

ในไมโครโปรเซสเซอร์ Z-80 มีรีจีสเตอร์ใช้งานเฉพาะอย่างซึ่งประกอบด้วย PC,SP,IX,IY,I และ R ซึ่งรีจีสเตอร์ต่าง ๆ เหล่านี้จะได้กล่าวถึงต่อไป

<u>โปรแกรมเคาเตอร์</u> (PC)

เป็นรีจีสเตอร์ขนาด 16 บิต มีหน้าที่เก็บตำแหน่งต่อไปที่จะถูกเฟทช์ออกมาจากหน่วยความจำ คำสั่งของซีพียู Z-80 เป็นคำสั่งที่มีความยาว 1,2,3 หรือ 4 ไบต์ ดังนั้นข้อมูลใน PC อาจถูกเพิ่มขึ้นครั้งละ 1,2,3 หรือ 4 โดยอัตโนมัติ ขึ้นอยู่กับความยาวของคำสั่งที่ถูกปฏิบัติก่อนหน้านั้น ถ้าคำสั่งที่มีความยาวต่าง ๆ กัน ถูกวางไว้เป็นลำดับดังรูปที่ 12 ค่าใน PC หลังจากที่คำสั่งนั้นถูกปฏิบัติไปแล้วคือค่าที่อยู่ทางด้านขวา



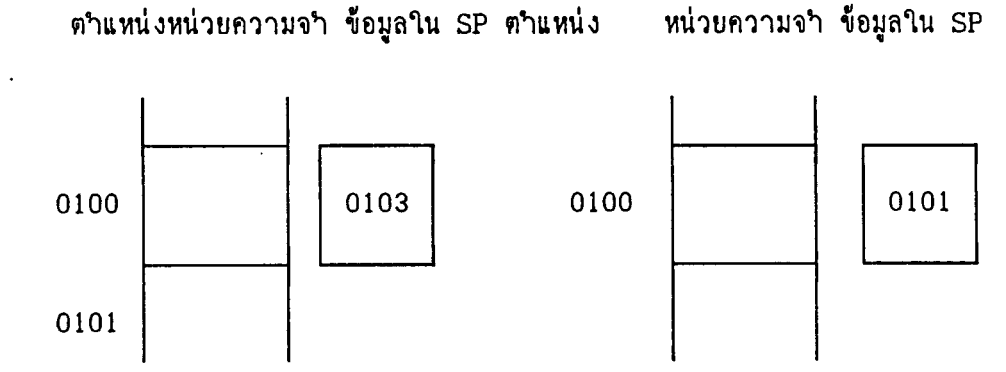


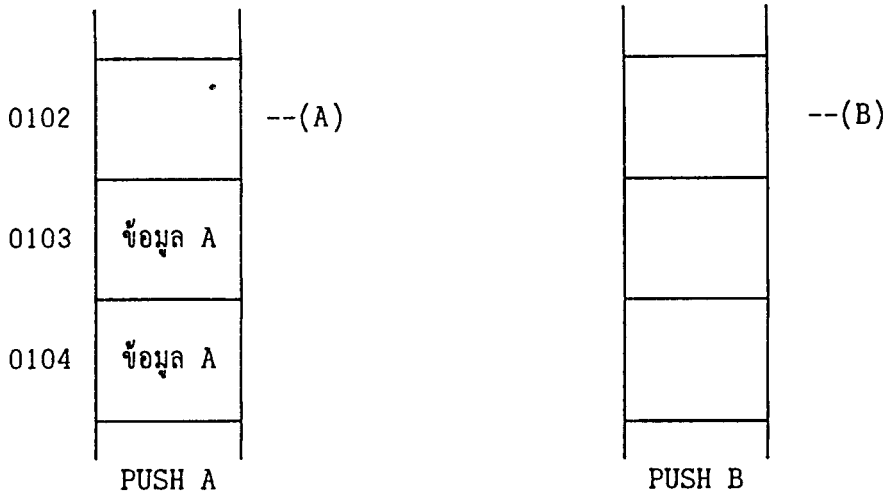
รูปที่ 12 แสดงการเพิ่มของข้อมูลในโปรแกรมเคาเตอร์

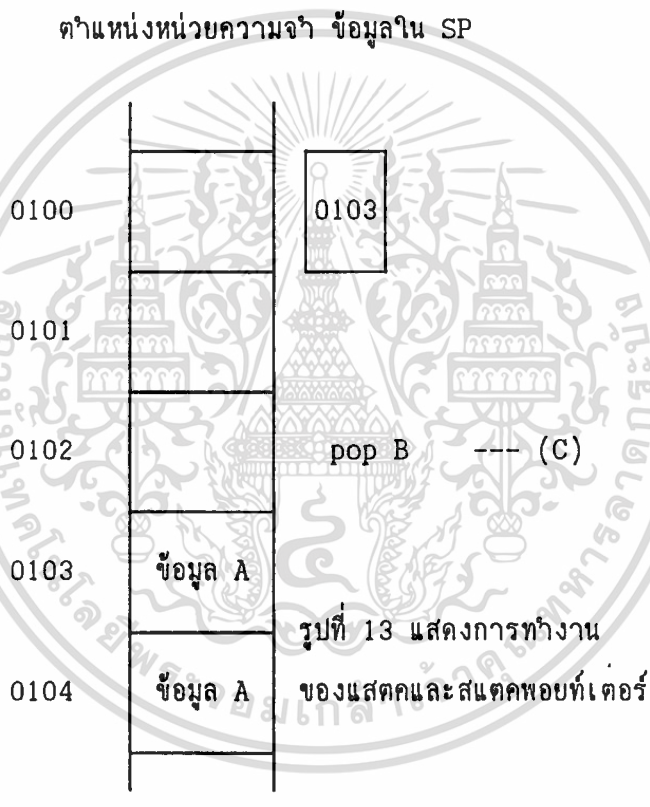
จะเห็นได้ว่าค่าของ PC จะเพิ่มขึ้นตามลำดับเรื่อย ๆ นอกจากว่าเมื่อ ซีพียู ทำคำสั่งในกลุ่มทำงานข้าม (Jump) มันจะเปลี่ยนไปตามตำแหน่งที่ต้องกระโดดไป โดยคำสั่งข้ามนั้น ๆ และอีกกรณีหนึ่งคือ เมื่อ ซีพียูมีการ Interupt ก็จะทำให้ค่าของ PC ไม่เรียงลำดับเช่นกัน และข้อมูลใน PC ไม่สามารถนำมากระทำทางคณิตศาสตร์และลอจิกได้

<u>สแตคพอยท์เตอร์</u> (SP)

สแตคพอยท์ของซีพียู Z-80 เป็นรีจีสเตอร์ที่มีขนาด 16 บิต มีหน้าที่ทำหน้าที่ใช้ชี้ตำแหน่งบนสุดของสแตค ซึ่งสแตคของไมโครโปรเซสเซอร์ Z-80 เป็นสแตคแบบใช้หน่วยความจำภายนอก (External menory stack) โดยใช้ส่วนหนึ่งของแรม (RAM) ซึ่งหน้าที่ของสแตคนี้คือใช้เก็บข้อมูลชั่วขณะ ซึ่งข้อมูลอาจมาจากรีจีสเตอร์ต่าง ๆ ข้อมูลที่เก็บจากสแตคนี้ อาจเก็บเข้าหรือนำออกโดยอัตโนมัติ เช่นเมื่อซีพียูทำคำสั่ง CALL หรือเมื่อซีพียูถูกขัดจังหวะมักจะเก็บค่าของ PC เข้าในสแตคโดยอัตโนมัติ และเมื่อซีพียูทำคำสั่ง RET RETI หรือ RETN ข้อมูลของ PC ที่เก็บไว้ในสแตคก็จะนำออกมาโดยอัตโนมัติเช่นกัน และนอกจากนี้ข้อมูลจากรีจีสเตอร์ต่าง ๆ ยังสามารถนำเข้า หรือนำออกจากสแตคโดยใช้คำสั่งก็ได้ เมื่อใช้คำสั่ง PUSH qq จะให้นำข้อมูลในรีจีสเตอร์เข้ามาเก็บในสแตค ส่วนคำสั่ง POP qq เป็นการนำข้อมูลในสแตคออกมาใส่ไว้ในรีจีสเตอร์ที่อ้างอิง ซึ่งในทุกกรณีของการนำข้อมูลเข้าหรือออกในสแตคนี้ ตำแหน่งบนสุดของสแตคพอยท์เตอร์เสมอ ตัวอย่างดังรูปที่ 13







รูปที่ 13 แสดงการทำงาน
ของแสตคและสแตคพอยท์เตอร์

สังเกตุได้ว่าข้อมูลที่อยู่ใน SP คือตำแหน่งสูงของสแตคที่ข้อมูลอยู่เสมอ

<u>อินเด็กซ์รีจีสเตอร์</u> (Index register)

ไมโครโปรเซสเซอร์ Z-80 มีอินเด็กรีจีสเตอร์ 16 บิต 2 ตัวคือ IX,IY ซึ่งทั้งสองตัวนี้มีลักษณะการทำงานที่เหมือนกันทุกประการ คือ ใช้ในคำสั่งที่อ้างถึงข้อมูลแบบอินเด็กซ์โดยที่ตำแหน่งของข้อมูลใช้ (Effeective address) ที่อยู่ในหน่วยความจำ กำหนดโดยนำข้อมูลที่อยู่ในอินเด็กซ์รีจีสเตอร์ ที่ใช้เป็นตำแหน่งฐาน (Base address) บวกกับระยะห่าง

(Displacement)ขนาด 8 บิต ที่บรรจุในส่วนหนึ่งของคำสั่ง เขียนเป็นสมการได้คือ

Effective address = Base address + Displacement

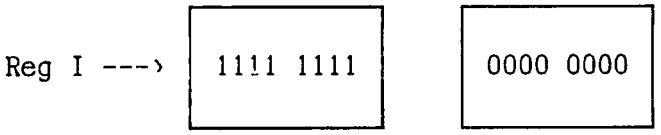
<u>อินเตอร์รัพต์เวคเตอร์รีจีสเตอร์</u> (I)

รีจีสเตอร์ขนาด 8 บิต ที่สามารถโหลดข้อมูลขนาด 8 บิต ลงไปได้ข้อมูลในรีจีสเตอร์ I นี้เป็นส่วนหนึ่งของตำแหน่งของหน่วยความจำที่อ้างถึง โดยใช้เป็นข้อมูล 8 บิตด้านสูง (High byte) เพื่อใช้ชี้ตำแหน่ง ของตารางเวคเตอร์ (Vector) และเมื่อนำข้อมูลอีก 8 บิต ที่ใช้ชี้ตำแหน่งหน่วยความจำในตารางเวคเตอร์ ข้อมูล 8 บิตางด้านต่ำที่จะนำมาต่อเป็นข้อมูลส่งมาจากอุปกรณ์อินพุท/เอาท์พุท เมื่ออุปกรณ์อินพุท/เอาท์พุท ส่งสัญญาณเพื่อทำการอินเตอร์รัพต์ การทำงานตามปกติของซีพียู และถ้าซีพียูตอบรับการอินเตอร์รัพต์นั้นอุปกรณ์ตัวที่ส่งสัญญาณการอินเตอร์รัพต์จะส่งข้อมูลขนาด 8 บิตมาให้ ซึ่งข้อมูลขนาด 8 บิต ทางด้านต่ำมารวมกับรีจีสเตอร์ I เพื่อชี้ตำแหน่งตารางเวคเตอร์ และข้อมูลที่อยู่ในหน่วยความจำตามตำแหน่งในตารางที่อ้างถึง 2 ไบต์ จะเป็นตำแหน่งที่เริ่มต้นของโปรแกรมบริการอินเตอร์รัพต์ของตัวอุปกรณ์นั้น ๆ ลักษณะการทำงานดังกล่าวนี้ อธิบายได้ดังรูปที่ 14 จากรูปกำหนดข้อมูลในรีจีสเตอร์ I คือ 02H เมื่อข้อมูลทั้งสองรวมกันได้ FF02H ซึ่งเป็นตำแหน่งของหน่วยความจำ ภายนอกที่อ้างถึงหน่วยความจำในส่วนนี้ เรียกว่าตารางเวคเตอร์ โดยตารางนี้จะบรรจุตำแหน่งของโปรแกรมบริการ การอินเตอร์รัพต์ไว้ ดังนั้นค่าของรีจีสเตอร์ I หนึ่งค่าสามารถกำหนดตำแหน่งการบริการของ การอินเตอร์รัพต์ได้ถึง 128 ตำแหน่ง จากรูปที่ 14 ตำแหน่งที่ FF02H และ FF03H จะบรรจุตำแหน่ง เริ่มต้นของโปรแกรมบริการอุปกรณ์ที่ส่งสัญญาณ การอินเตอร์รัพต์เข้ามา ดังนั้น โปรแกรมบริการของอุปกรณ์จึงอยู่ที่ตำแหน่ง B000H รีจีสเตอร์ I นี้ ใช้เพื่อทำการอินเตอร์รัพต์ในโหมด 2 ซึ่งการอินเตอร์รัพต์ของซีพียู Z-80 ซึ่งมี 3 โหมด

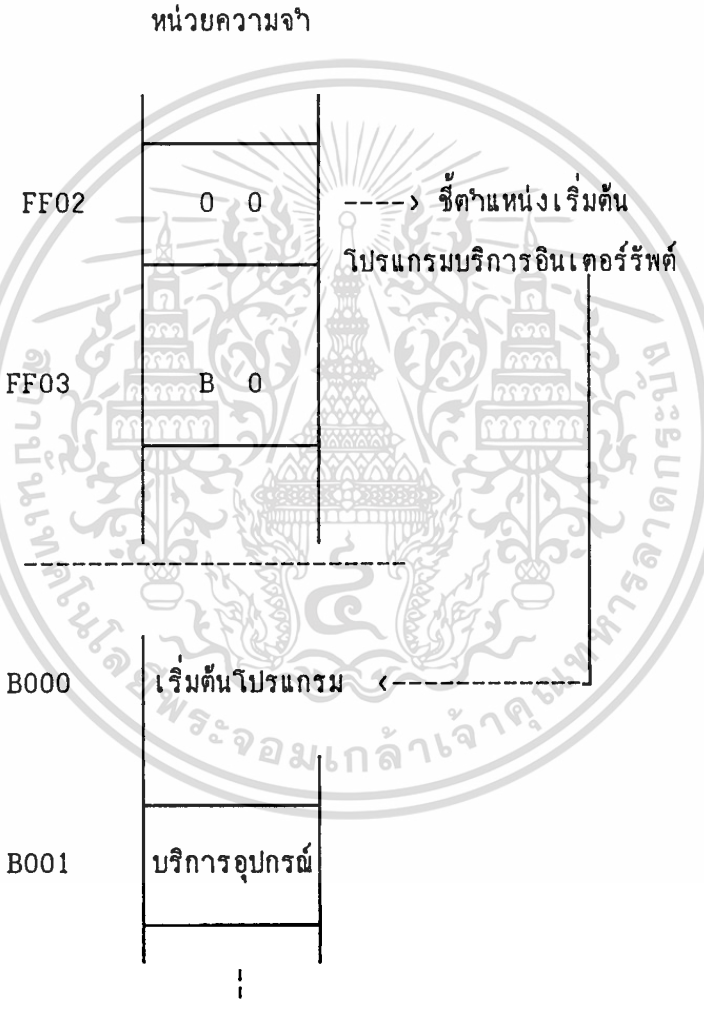
<u>รีเฟรสรีจีสเตอร์</u> (R)

เมื่อซีพียู Z-80 ต่อหน่วยความจำแบบไดนามิค ซึ่งหน่วยความจำแบบนี้จะต้องมีการอัดซ้ำ (Refresh) ตลอดเวลา (ปกติทุก ๆ 2 mS)เพื่อให้ข้อมูลในหน่วยความจำนี้คงอยู่ ซีพียู Z-80 นี้ใช้รีจีสเตอร์ R เพื่อบรรจุตำแหน่งของหน่วยความจำ เพื่อทำการอัดซ้ำ รีจีสเตอร์ R มี ขนาด 7 บิต และข้อมูลในรีจีสเตอร์จะเพิ่มขึ้นทีละ 1 โดยอัตโนมัติหลังจากที่เฟทต์คำสั่งหนึ่ง ๆ แล้วข้อมูลใน

รีเฟรชรีจีสเตอร์ ถูกส่งออกไปทางด้านล่างของบัสตำแหน่ง (A6-A0)ในขณะที่สัญญาณรีเฟรชแอคตีฟ และซีพียูไม่ได้ Access หน่วยความจำรีจีสเตอร์ R โดยปกติผู้เขียนโปรแกรมไม่จำเป็นต้องใช้นอกจากการตรวจสอบเท่านั้น



ข้อมูล 16 บิต ระบุตำแหน่งหน่วยความจำ : FF02H

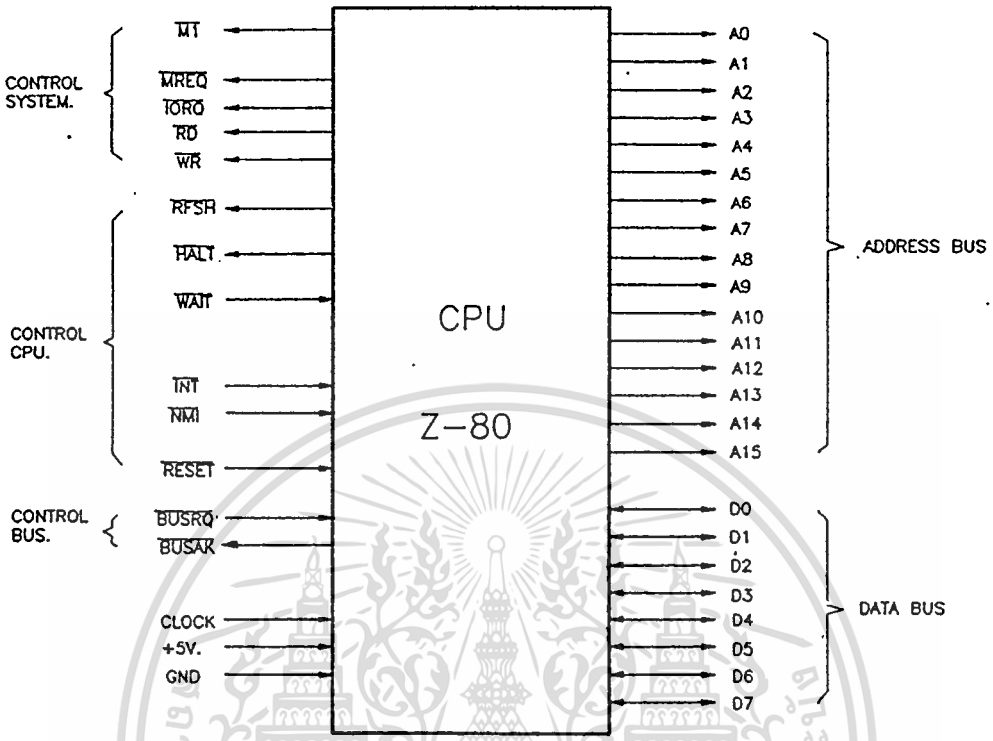


รูปที่ 14 การทำงานของรีจีสเตอร์ I

ขาและสัญญาณการเชื่อมต่อ Z-80

ไมโครโปรเซสเซอร์ Z-80 อยู่ในไอโอชีขนาดมาตรฐานอุตสาหกรรม (In dustry Standard) แบบ Dual In-Line Package (DIP) หรือที่เรียกว่าแบบตีนตะขาบ 40 ขา ๆ

ต่าง ๆ แสดงไว้รูปที่ 15 กลุ่มสัญญาณต่าง ๆ ของไมโครโปรเซสเซอร์แบ่งออกได้เป็น 3 กลุ่ม คือ



รูปที่ 15 แสดงขาต่าง ๆ ของซีพียู Z-80

กลุ่มของสายสัญญานเพื่อกำหนดตำแหน่ง (Address Bus) คือ A-15-A0 กลุ่มของสายสัญญาณข้อมูล (Data Bus) คือสัญญาณที่เหลือยกเว้นขาแหล่งจ่ายไฟและสัญญาณนาฬิกา หน้าที่ของขาต่าง ๆ จะได้อธิบายในรายละเอียดต่อไปนี้

A15-A0 เป็นสายสัญญาณกำหนดตำแหน่ง (Address-Bus)โดยที่ A0 เป็นบิตทางต่ำ (LSB) ขาเหล่านี้เป็นเอาท์พุทแบบสามสถานะ (Tri-State) แอคตีฟที่ลอจิก 1 บัสนี้มีด้วยทั้งหมด 16 สาย ดังนั้น จึงสามารถติดต่อกับหน่วยความจำได้ถึง 2^{16} = 65536 ตำแหน่ง (64 Kbyte) นอกจากนั้นยังสามารถใช้การกำหนดตำแหน่งของอินพุท/เอาท์พุทเมื่อใช้คำสั่งอินพุท/เอาท์พุทได้โดยใช้ 8 บิต ด้านต่ำ (A7-A0) เพื่อแสดงตำแหน่งของพอร์ต ดังนั้นสามารถกำหนดพอร์ตอินพุทได้ 256 พอร์ตเช่นกันและช่วงขณะเวลารีเฟรช

(RFSH) บัสทางต่ำ 7 บิต (A6-A0) จะใช้แสดงตำแหน่งของหน่วยความจำแบบไดนามิคที่ได้รับจากรีเฟรช

D7-D0 เป็นสายสัญญาณข้อมูล (Data Bus) D0-D7 เป็นบิตทางต่ำลักษณะเป็นบัสข้อมูลแบบสามสถานะ ขนาด 8 บิตและแอทีฟที่ลอจิก 1ใช้เพื่อเป็นเส้นทางผ่านของข้อมูลระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำหรืออุปกรณ์อินพุท/เอาท์พุทต่าง ๆ

$\overline{M1}$ (Machine Crycle One) เป็นขาเอาท์พุทและแอคตีฟที่ลอจิก 0 เมื่อขานี้แอคตีฟ ชี้ให้เห็นว่าขณะนี้กำลังอยู่ในสภาวะของการเฟทช์คำสั่ง และถ้าเป็นคำสั่งที่มีรหัส 2 ไบต์ ส่วนของ $\overline{M1}$ จะถูกสร้างขึ้นขณะในเฟทช์แต่ละไบต์ ลักษณะของคำสั่งที่มีขนาด 2 ไบต์ เช่นคำสั่งที่มีรหัสเริ่มต้นด้วย CBR,DDH,EDH, หรือ FDH นอกจากนั้นสัญญาณ $\overline{M1}$ นี้จะใช้ร่วมกับ เพื่อสร้าง $\overline{IORQ}$ สัญญาณตอบรับการอินเตอร์รัพธ์ (Interrupt Acknowledge)

$\overline{MREQ}$ (Memory Request) เป็นเอาท์พุตแบบสถานะและแอคทีฟที่ลอจิก 0 เมื่อสายสัญญาณนี้แอคทีฟ บอกให้ทราบว่า ขณะนี้ไมโครโปรเซสเซอร์ต้องการติดต่อกับหน่วยความจำเพื่ออ่านหรือเขียนข้อมูลโดยที่ตำแหน่งของหน่วยความจำจะปรากฏอยู่บัสตำแหน่งแล้ว

$\overline{IORQ}$ (Input/Request) เป็นเอาท์พุทแบบสามสถานะ และแอคตีฟที่ลอจิก 0 เมื่อสัญญาณแอคตีฟบอกให้ทราบว่า ขณะนี้ทางด้านไบต์ต่ำกว่า (A7-A0) ของบัสตำแหน่งบรรจุตำแหน่งของพอร์ต ที่จะถ่ายข้อมูลระหว่างไมโครโปรเซสเซอร์กับอุปกรณ์ อินพุท/เอาท์พุท นอกจากนี้จะช่วยร่วมสัญญาณ M1 เพื่อตอบรับการอินเตอร์รัพต์ และ ขณะนี้ เวกเตอร์ของอินเตอร์รัพต์จะส่งผ่านเข้ามาในบัสข้อมูลเพื่อกำหนดตำแหน่งของโปรแกรมบริการการอินเตอร์รัพต์

$\overline{RD}$ (Menory Read) เป็นขาเอาท์พุทแบบสามสถานะ และแอคตีฟที่ลอจิก 0 สัญญาณนี้เพื่อชี้ว่าขณะนี้ ไมโครโปรเซสเซอร์ต้องการอ่านข้อมูลจากหน่วย

ความจำหรือจากอุปกรณ์อินพุท/เอาท์พุท

$\overline{RFSH}$ (Menory Write) เป็นขาเอาท์พุทแบบสามสถานะ และแอคตีฟที่ลอจิก 0 เมื่อสัญญาณแอคตีฟชี้ว่าขณะนี้ บัสตำแหน่งด้านต่ำ 7 บิต (A6-A0) บรรจุตำแหน่งหน่วยความจำแบบไดนามิคแรมที่จำรีเฟรชและสัญญาณ $\overline{MERQ}$ ในช่วงนี้จะนำไปใช้เป็นสัญญาณสำหรับเพื่อรีเฟรช (Refresh Read) ไดนามิคแรมทั้งหมดที่ใช้ในระบบ

$\overline{HALT}$ (Halt State)เป็นขาเอาท์พุทแอคตีฟที่ลอจิก 0 เป็นสัญญาณเพื่อชี้ว่าขณะนี้ไมโครโปรเซสเซอร์ปฏิบัติคำสั่ง $\overline{HALT}$ จากโปรแกรมและกำลังรอสัญญาณการอินเตอร์รัพต์ชนิดนอนมาสเคเบิล (เมื่อสั่งให้ยอมรับ) จากอุปกรณ์ภายนอก ถ้าได้รับสัญญาณการอินเตอร์รัพต์แล้วจึงจะทำงานต่อไปได้ในขณะที่หยุดรอ (Halted) นี้ซีพียูจะกระทำคำสั่ง NOP (No-Operation) เพื่อให้มีการเฟทช์คำสั่งซึ่งจะไม่ทำให้การรีเฟรชหยุดชงักลง

$\overline{WAIT}$ (Wait) เป็นขาอินพุท แอคตีฟที่ลอจิก 0 เป็นสัญญาณเพื่อชี้ว่าการส่งถ่ายข้อมูลระหว่างไมโครโปรเซสเซอร์และหน่วยความจำ หรืออุปกรณ์อินพุท/เอาท์พุทยังไม่เรียบร้อย และให้ไมโครโปรเซสเซอร์ หยุดรอ ตราบเท่าที่ขานี้ยังแอคตีฟอยู่ ดังนั้นสัญญาณนี้จะใช้เพื่อให้หน่วยความจำหรืออุปกรณ์อินพุท/เอาท์พุท ที่มีความเร็วใด ๆ สามารถให้เข้าจังหวะกันพอดี (Synchronized) กับไมโครโปรเซสเซอร์

$\overline{INT}$ (Interrupt Request)เป็นขาอินพุทแอคตีฟที่ลอจิก 0 สัญญาณ $\overline{INT}$ นี้เป็นสัญญาณที่สร้างมาจากอุปกรณ์อินพุท/เอาท์พุทเพื่อต้องการอินเตอร์รัพต์การทำงานตามปกติของไมโครโปรเซสเซอร์จะจดจำไว้ ถ้าหากว่าโปรแกรมกำหนดให้มีการยอมรับสัญญาณ การอินเตอร์รัพต์ได้ (Enable Interupt) โดย IFF ถูกเซตเป็น 1 และไม่มีการขอใช้บัสเสียก่อน คือขา $\overline{BUSRQ}$ ต้องไม่แอคตีฟ เมื่อไมโครโปรเซสเซอร์รับสัญญาณอินเตอร์รัพต์ มันจะตอบสนองโดยการตอบสัญญาณ $\overline{IORQ}$ ออกมาในช่วงเวลา $\overline{M1}$ เพื่อเป็นการตอบรับการอินเตอร์รัพต์ (Interrupt Acknowleade) ใน

ช่วงไซเกิลของคำสั่งต่อมา

$\overline{NMI}$ (Non Maskable Interrupt) เป็นขาอินพุทและแอคตีฟที่ขอบพัลส์ขาลง (Negativedge Trigger) สัญญาณที่ขา $\overline{NMI}$ นี้ ลำดับความสำคัญสูงกว่าสัญญาณที่ขา $\overline{INT}$ ไมโครโปรเซสเซอร์ จะทำ การตรวจสอบที่ขานี้ที่สเตทสุดท้ายของคำสั่ง เช่นเดียวกับขา $\overline{INT}$ แต่ไม่ขึ้นอยู่กับ IFF เมื่อไมโครโปรเซสเซอร์ได้รับสัญญาณที่ขา $\overline{NMI}$ จะทำให้เริ่มต้นการทำงานใหม่ที่ต่ำกว่าตำแหน่ง 0066H ส่วนค่าในโปรแกรมเคาน์เตอร์ที่ชี้ตำแหน่ง ของคำสั่งต่อไปก่อนที่ ซีพียู จะถูกอินเตอร์รัพต์ จะเก็บไว้ในสแตค (ที่ RAM) เพื่อที่ซีพียูสามารถกลับมาทำงานต่อไปหลังจากที่ทำโปรแกรมบริการการอินเตอร์รัพต์เสร็จสิ้นแล้ว ในขณะที่ซีพียูอยู่ในจังหวะ Wait มันจะไม่รับสัญญาณ $\overline{NMI}$ ลำดับความสำคัญต่ำกว่าสัญญาณ $\overline{BUSRQ}$ ดังนั้นในขณะที่ซีพียูกำลังทำโปรแกรมการบริการการอินเตอร์รัพต์อยู่ มันสามารถรับสัญญาณ BUSRQ ได้

$\overline{RESET}$ เป็นอินพุท แอคตีฟที่ลอจิก 0 เมื่อไมโครโปรเซสเซอร์ ได้รับสัญญาณ $\overline{RESET}$ จะทำให้คำในโปรแกรมเคาน์เตอร์เริ่มต้นที่ศูนย์ และตั้งต้นการทำงานของไมโครโปรเซสเซอร์ใหม่ และ ในส่วนต่าง ๆ จะเป็นดังนี้

1. จัดอินเตอร์รัพต์ฟลิบฟลอป (IFF)ให้อยู่ในสภาพที่ไม่ยอมรับการอินเตอร์รัพต์แบบมาสเคเบิล (IFF1=IFF2=0)
2. เซตรีจีสเตอร์ I=00H
3. เซตรีจีสเตอร์ R=00H
4. เซตเป็นการอินเตอร์รัพต์ โหมด 0

ในช่วงเวลาของการรีเซต บัสข้อมูล บัสตำแหน่ง จะอยู่ในสภาวะอิมพีแดนซ์สูง ส่วนบัสควบคุมจะอยู่ในสภาพที่ไม่แอคตีฟ (Inactive)

$\overline{BUSRQ}$ (Bus Requsst) เป็นขาอินพุทแอคตีฟที่ระดับ 0 สัญญาณ $\overline{BUSRQ}$ นี้มีผลทำให้บัสตำแหน่งข้อมูล และสัญญาณควบคุมที่เป็นขาเอาท์พุทแบบสามสถานะอยู่ในสภาวะอิมพีแดนซ์สูงจากนั้นต่าง ๆ จะถูกควบคุมโดยอุปกรณ์ภายนอกไมโครโปรเซสเซอร์ จะตรวจสอบการขอสัญญาณใช้บัสนี้ทุก ๆ สเตทสุดท้ายของแมชชีนไซเกิลของคำสั่ง เมื่อพบการขอใช้บัสซีพียูจะตอบสนองในไซเกิลถัดไป

$\overline{\text{BUSAK}}$ (Bus Acknowledge) เป็นเอาท์พุทแอคตีฟที่ระดับ 0 สัญญาณนี้ ใช้สำหรับตอบรับการขอใช้บัสและแสดงว่าขณะนี้บัสตำแหน่งข้อมูล และ สัญญาณควบคุมที่เป็น เอาท์พุทแบบสามสถานะอยู่ในสภาวะอิมพีแดนซ์สูงแล้ว อุปกรณ์ควบคุมภายนอก สามารถเข้าควบคุมบัสได้เป็นขาที่รับสัญญาณ นาฬิกาซึ่งเป็นเพียงเฟสเดียวกัน ใช้ระดับสัญญาณแบบ TTL และต้องการตัวต้านทานเพื่อ Pull up ค่า 330 โอห์ม หนึ่งตัวเพื่อต่อกับแหล่งจ่ายไฟ 5 โวลท์ Z-80 ทำงานได้ที่ สัญญาณนาฬิกาไม่เกิน 2.5 MHz Z-80A ทำงานได้ไม่เกิน 4 MHz และ Z-80B ทำงานไม่เกิน 6 MHz

ไดอะแกรมเวลาของไมโครโปรเซสเซอร์

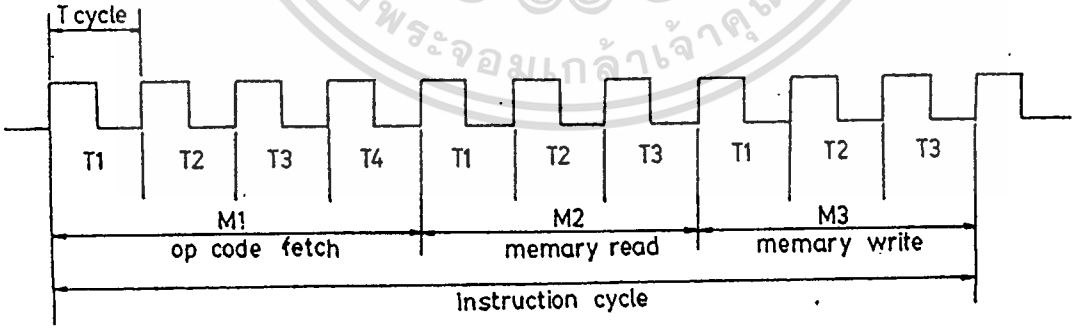
ไมโครโปรเซสเซอร์ Z-80 จำปฏิบัติคำสั่งต่าง ๆ อย่างมีขั้นตอนที่แน่นอนขั้นตอนเหล่านี้ประกอบด้วยการทำงานพื้นฐานต่าง ๆ คือ

-การอ่าน-เขียนหน่วยความจำ (Memory read or write)

-การอ่าน-เขียนอุปกรณ์อินพุท/เอาท์พุท (I/O device read or write)

-การบอกรับอินเตอร์รัพต์ (Interrupt Acknowledge)

การทำงานของคำสั่ง ต่าง ๆ เกิดจากลำดับการทำงานพื้นฐานเหล่านี้ขบวนการทำงานพื้นฐานแต่ละส่วนอาจใช้เวลา 3-6 คาบ กว่าจะเสร็จสมบูรณ์ หรืออาจขยายออกไปได้มากกว่านี้ เพื่อให้การทำงานไมโครโปรเซสเซอร์พอดีกับความเร็วของอุปกรณ์ภายนอก



รูปที่ 16 ตัวอย่างไดอะแกรมเวลาของไมโครโปรเซสเซอร์

คาบเวลาของสัญญาณหนึ่ง ๆ นี้คือ T ไซเกิล และขบวนการพื้นฐานหนึ่ง ๆ จะเรียกว่า

แมชชีนไซเกิล และขบวนการพื้นฐานต่าง ๆ จะเรียกว่า แมชชีนไซเกิล (Machine Cycle) ตัวอย่างไดอะแกรมเวลาของคำสั่งหนึ่งซึ่งแสดงดังที่รูป 16 จากรูป 16 เป็นคำสั่งประกอบด้วย 3 แมชชีนไซเกิล (M1,M2 และ M3) แมชชีนไซเกิลที่ 1 (M1) ของคำสั่งใดๆเป็นไซเกิลของการเฟทช์ ซึ่งอาจมี 4,5 หรือ 6T ไซเกิล ยกเว้นในการที่เพิ่มเวลาออกไปเนื่องจากมีการรอคอย (Wait state) ไซเกิลของการเฟทช์ (M2) เพื่อใช้ การอ่านรหัสคำสั่งต่อไปที่จะกระทำลำดับต่อมาทำการเคลื่อนย้ายข้อมูล ระหว่างไมโครโปรเซสเซอร์ กับหน่วยความจำหรืออุปกรณ์ อินพุท/เอาท์พุท ซึ่งใช้เวลาตั้งแต่ 3 ถึง 5 คาบเวลา เว้นแต่การรอคอยเช่นกัน แมชชีนไซเกิลที่ตามหลัง M1 นี้จะมีหรือไม่ก็แล้วแต่ชนิดของคำสั่ง เช่น LD r,r จะใช้เพียงหนึ่งแมชชีนไซเกิลก็สามารถทำงานได้ ต่อไปจะได้กล่าวถึงไดอะแกรมเวลาซึ่งปรากฏอยู่ในการทำงานพื้นฐานต่าง ๆ ทั้งที่ช่วงการรอคอยไม่มีการรอคอย ซึ่งแมชชีนไซเกิลที่มีใช้ สามารถสรุปได้ดังนี้

1. ไซเกิลการเฟทช์รหัส (Instruction Opcode fetch cycle(M1))

2.ไซเกิลการอ่าน หรือ เขียนหน่วยความจำ (Memory dataread or write cycle)

3.ไซเกิลการอ่านหรือเขียนข้อมูลกับอุปกรณ์ อินพุท/เอาท์พุท (I/O read or write cycle)

4.ไซเกิลการขอใช้บัสหรือการตอบสนองการขอใช้บัส (Bus request/acknowled cycle)

5.ไซเกิลการขออินเตอร์รัพต์ และการตอบรับ การอินเตอร์รัพต์ (Interrupt request/acknowledge cycle)

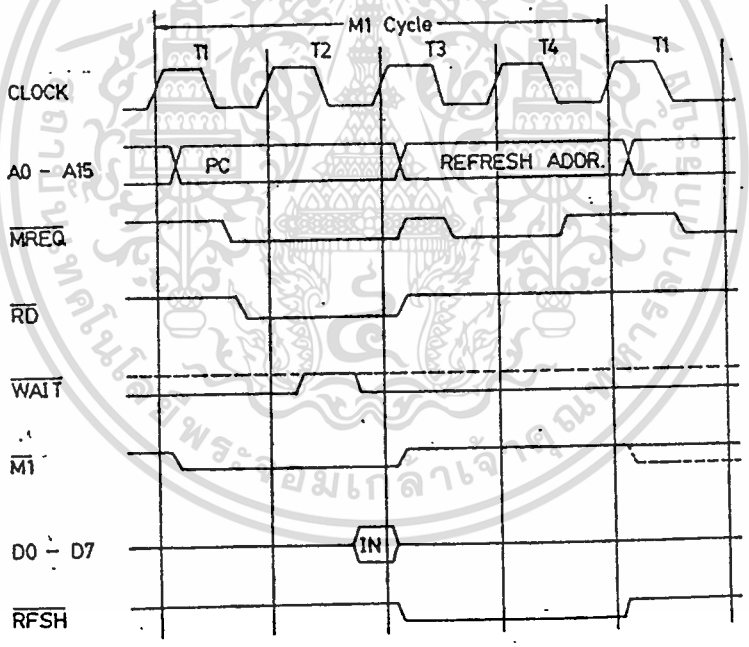
6.ไซเกิลการขอและตอบรับการอินเตอร์รัพต์แบบนอนมาสเคเบิล (Non-maskable interrupt request/acknowledge cycle)

7. ไซเกิลการออกคำสั่ง HALT (Exit from a HALT instruction) ต่อไปจะได้อธิบายการทำงานในไซเกิลต่าง ๆ

<u>ไซเกิล เฟ็ทช์คำสั่ง</u>

ไดอะแกรมของช่วงไซเกิลการเฟทช์คำสั่ง (M1 cycle) แสดงดังรูปที่ 17 M1 นี้เป็นสภาวะเริ่มแรกของทุกคำสั่ง สังเกตุได้ว่าเมื่อเริ่มไซเกิล M1 ในโปรแกรมเคาเตอร์ จะแทนที่ลงในบัสตำแหน่ง และหลังจากไซเกิลของคาบสัญญาณนาฬิกาต่อมา ซีพียูจะส่งสัญญาณ $\overline{\text{MREQ}}$ ออกมาที่เวลานี้ตำแหน่งของหน่วยความจำบนบัสตำแหน่งจะคงที่ ดังนั้นช่วงขอบพัลส์ขาลงของ $\overline{\text{MREQ}}$ จะใช้เป็นสัญญาณ chip enable ของหน่วยความจำในขณะนี้สัญญาณ $\overline{\text{RD}}$ แอคตีฟ เพื่อแสดงว่า ซีพียู

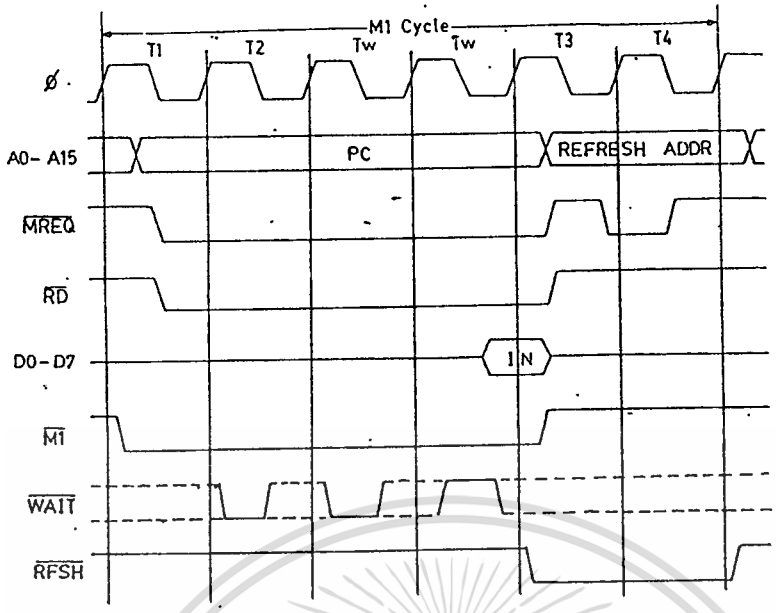
ต้องการอ่านข้อมูลจากหน่วยความจำ ตามตำแหน่งที่ปรากฏบนบัส ตำแหน่ง ดังนั้นหน่วยความจำภายนอกจะส่งข้อมูลจากตำแหน่งที่ระบุมาบนบัสข้อมูล ซีพียูจะสุ่มอ่านข้อมูลจากหน่วยความจำบนบัสข้อมูลเมื่อถึงจังหวะขอบพัลส์ขาขึ้นของ T3 ขอบพัลส์นี้จะใช้ในการยกเลิกสัญญาณ $\overline{MREQ}$ และ $\overline{RD}$ ด้วยดังนั้นจะเห็นว่าข้อมูลที่ถูกอ่านเข้าใน IR ก่อนที่สัญญาณ $\overline{MREQ}$ และ $\overline{RD}$ จะหายไปตามเวลา T2 และ T4 ที่เหลือในไซเกิล M2 จะใช้ในการรีเฟรชหน่วยความจำแบบไดนามิก (ในขณะเดียวกัน ซีพียูก็ใช้เวลาช่วงนี้ถอดรหัส และปฏิบัติคำสั่ง ในกรณีที่คำสั่งนั้น ๆ ไม่ต้องการแมชชีนไซเกิลอื่น ๆ อีก เช่นคำสั่ง INC r เป็นต้น) ขณะเวลา T3 และT4 นี้บัสตำแหน่งทางด้านต่ำ 7 บิต (A6-A0) จะบรรจุด้วยตำแหน่งของหน่วยความจำที่จะทำการรีเฟรช และสัญญาณรีเฟรชจะแอคตีฟ เพื่อ แสดงว่าการอ่านเพื่อรีเฟรช (Refresh read) สำหรับไดนามิกแรมทุกตัวควรจะทำในขั้นตอนนี้ให้เสร็จและสังเกตุว่าขณะที่ทำการรีเฟรชจะไม่มีการกำเนิดสัญญาณ $\overline{RD}$ เพื่อป้องกันไม่ให้ หน่วยความจำตำแหน่งอ่านเข้ามาในบัสข้อมูลสัญญาณ $\overline{MREQ}$ ในช่วงเวลาการรีเฟรชใช้เพื่อการอ่านเพื่อการรีเฟรชทุก ๆ ตำแหน่งของหน่วยความจำ



รูปที่ 17 ไซเกิล M1 (Opcode cycle)

ไซเกิลการเฟทช์คำสั่งที่มีการรอคอย

สถาวะการรอคอย (Wait state) การเกิดเนื่องจากความเร็วในการทำงานของหน่วยความจำ ช้ากว่าการทำงานของซีพียู ดังนั้นการเฟรชคำสั่งก็จะช้าลง พิจารณาจากรูปที่ 18ในช่วงขอบพัลส์ขาลงของ T2 และทุก ๆ TW ที่ตามมา ซีพียูจะทำการตรวจสอบซึ่งขา $\overline{WAIT}$ แอคตีฟ



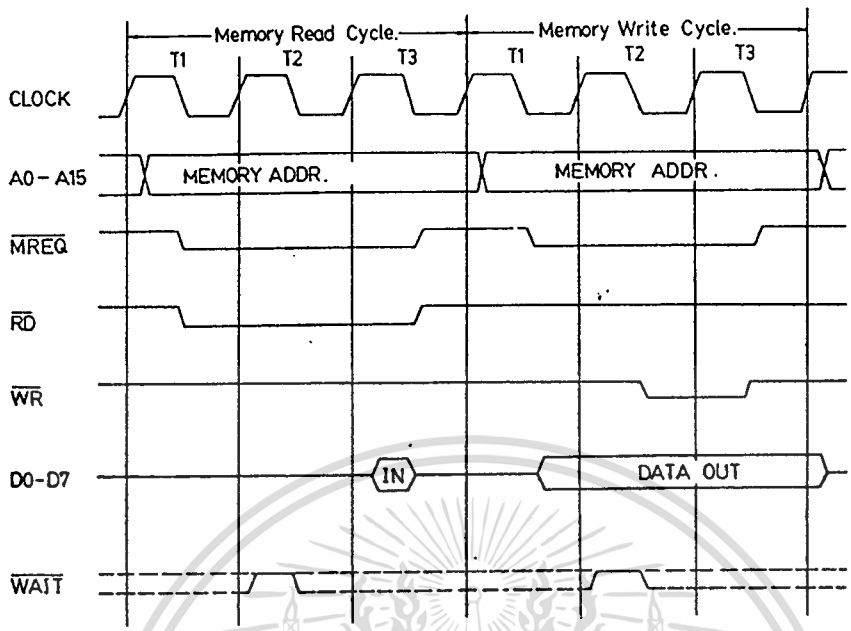
รูปที่ 18 ไซเกิล M1 ที่มีช่วงการรอคอย

ในช่วงนี้ซีพียูจะสร้าง Tw เข้ามา 1 คาบ ของสัญญาณนาฬิกาถัดไป วิธีการแบบนี้ ทำให้สามารถหน่วงเวลาการอ่านข้อมูลให้ช้าลงเพื่อให้สอดคล้องกับ Access time ของหน่วยความจำแบบต่าง ๆ

ไซเกิล การอ่านหรือเขียน หน่วยความจำ

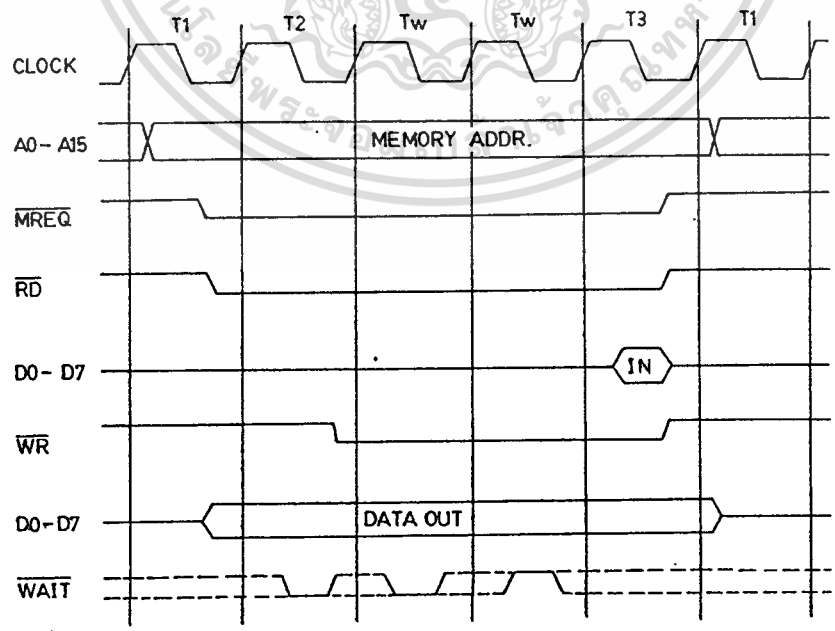
ไดอะแกรมของการอ่านหรือเขียนหน่วยความจำ แสดงดังรูปที่ 19 ไซเกิลการทำงานทั้งสองนี้ ใช้สัญญาณนาฬิกาอย่างละ 3 คาบ ไม่รวมคาบเวลาการรอคอยที่เกิดขึ้นที่ขา $\overline{WAIT}$ พิจารณาไซเกิลการอ่าน เห็นได้ว่าขั้นแรก ซีพียูจะส่งตำแหน่งที่ต้องการอ่านลงในบัสตำแหน่ง และหลังจากขอบพัลส์ขาลงของ T1 ก็ส่งสัญญาณ $\overline{MREQ}$ และ $\overline{RD}$ ออกมา (เหมือนช่วงการเฟทช์) จากนั้นหน่วยความจำจะส่งข้อมูลในตำแหน่งนั้นลงมาในบัสข้อมูล เมื่อถึงขอบพัลส์ขาลงของ T2 ซีพียูจะตรวจสอบที่ขา $\overline{WAIT}$ ว่ามีสัญญาณการรอคอยหรือไม่ ถ้ามี ซีพียูจะสร้าง Tw ขึ้น ในคาบเวลาต่อไป ถ้าไม่มีซีพียูจะทำงานต่อไปคือเมื่อถึงช่วงขอบพัลส์ขาลงของ T3 มันจะทำการอ่านข้อมูลจากบัสข้อมูลในระยะเวลาสั้น ๆ เข้าไปเพื่อไม่ให้เกิดความผิดพลาด ส่วนไซเกิลของการเขียนข้อมูลลงหน่วยความจำพิจารณาได้จากรูปที่ 19 ซึ่งลักษณะการทำงานจะคล้ายกับช่วงการอ่าน คือหลังจากส่งตำแหน่งที่ต้องการเขียนลงไปบัสตำแหน่งแล้วก็จะส่งสัญญาณ $\overline{MREQ}$ ออกมาจากนั้นข้อมูลจากรีจีสเตอร์ที่ต้องการเขียนลงไปในหน่วยความจำจะถูกถ่ายลงในบัสข้อมูลที่ขอบพัลส์ขาลงของ T1 และ ขอบพัลส์ขาลงของ T2 สัญญาณ $\overline{WR}$ จะแอคตีฟ ซึ่งในหน่วยความจำภายนอกเขียนข้อมูลเข้าไปในหน่วย

ความจำตามตำแหน่งที่กำหนดที่บัสตำแหน่งของพัลส์ขาลงของ T3 สัญญาณ $\overline{MREQ}$ และ $\overline{WR}$ จะหาย



รูปที่ 19 ไซเกิลการอ่านและเขียนหน่วยความจำ

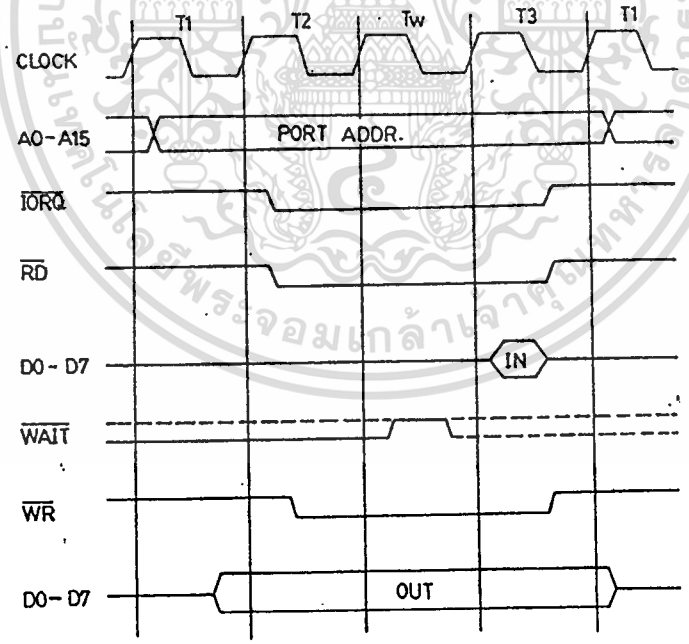
ไป หลังจากนั้นข้อมูลบนบัสข้อมูลก็สามารถเปลี่ยนแปลงต่อไปได้ แต่ถ้ามีสัญญาณการรอคอยเกิดขึ้นที่ขอบพัลส์ขาลงของ T2 ก็จะเกิด Tw ขึ้นต่อไปซึ่งเหมือนกันกับไซเกิล การเฟทช์คำสั่ง และไดอะแกรมเวลาเมื่อมีสัญญาน $\overline{WAIT}$ จะแสดงดังรูปที่ 20



รูปที่ 20 ไซเกิลการอ่านหรือเขียนหน่วนความจำเมื่อมีสัญญาณการรอคอย

ไซเกิลการอ่านหรือเขียนข้อมูลกับอุปกรณ์อินพุท/เอาท์พุท

ไซเกิลการอ่านหรือเขียนข้อมูลกับอุปกรณ์ อินพุท/เอาท์พุทนี้ เกิดขึ้นในขณะที่ ซีพียู ปฏิบัติคำสั่งเกี่ยวกับการอินพุทหรือเอาท์พุท เช่น IN A,(n) หรือ OUT (n), A เป็นต้น ปกติคำสั่งกลุ่มนี้ใช้ 3 หรือ 4 แมชชีนไซเกิล แต่คำสั่งที่มีความสามารถสูงในบางคำสั่ เช่น INTR INDR PTIR หรือ OTDR ที่สามารถส่งถ่ายข้อมูลได้ถึง 256 ไบต์ จะใช้แมชชินไซเกิลบางอันซ้ำ ๆ กันจนกระทั่งส่งถ่ายข้อมูลเสร็จผลก็คือ เวลาในการทำงานของคำสั่งจะขึ้นอยู่กับจำนวนไบต์ที่จะส่งถ่ายและความเร็วของอุปกรณ์ อินพุท/เอาท์พุท ด้วยไดอะแกรมเวลาของการอ่านข้อมูลจากอุปกรณ์ อินพุทและการเขียนข้อมูลลงในอุปกรณ์ เอาท์พุท แสดงได้ดังรูปที่ 21 จากรูปเห็นได้ว่า เมื่อเริ่มต้นแมชชีนไซเกิลตำแหน่งของ อุปกรณ์อินพุท/เอาท์พุท จะถูกแทนลงในบัสตำแหน่งทางด้านไบต์ต่ำ (A7-A0) หลังจากที่ขอบพัลส์ขาขึ้น ของ T2 สัญญาณ $\overline{\text{IORQ}}$ ตัวควบคุมอุปกรณ์ภายนอกจะรับรู้สัญญาณการขออ่านนี้และนำข้อมูลส่งไปบนบัสข้อมูลจากนั้นซีพียูจะสร้าง Tw ขึ้นมา 1 คาบเวลาทั้งๆ ที่ไม่มีสัญญาณ $\overline{\text{WAIT}}$ และ การตรวจสอบสัญญาณ $\overline{\text{WAIT}}$ จะทำให้ขอบพัลส์ขาลงของ T แทนการอ่านข้อมูลจากบัสข้อมูลเข้าซีพียูจะทำที่ขอบขาพัลส์ลงของ T3



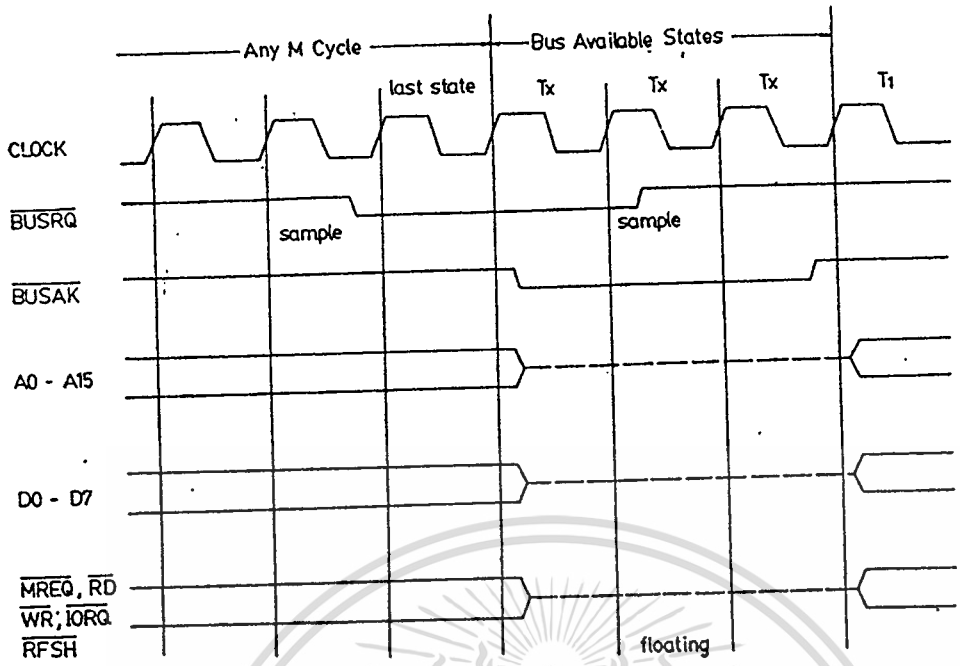
รูปที่ 21 ไซเกิลของการเขียนอุปกรณ์อินพุท/เอาท์พุท

ไซเกิลของการเขียนข้อมูล สัญญาณ $\overline{WR}$ จะแอคตีฟในเวลาเดียวกับสัญญาณ $\overline{IORQ}$ ดังรูปที่ 21 ข้อมูลจากรีจีสเตอร์ที่จะส่งไปยังอุปกรณ์ อินพุท/เอาท์พุท จะถูกถ่ายลงบัสข้อมูลที่ขอบพัลส์ขาลงของ T1 คือก่อนที่มีสัญญาณ $\overline{IORQ}$ และ $\overline{WR}$ หลังจาก T2 ซีพียูจะสร้าง Tw ขึ้นมาอัตโนมัติ 1 คาบ เช่นเดียวกับช่วงอ่าน และจากนั้นในช่วงเวลาที่เหลือคือ Tw ตัวควบคุมอุปกรณ์ อินพุท/เอาท์พุท จะนำข้อมูลจากบัสข้อมูลเข้าไป

จากการทำงานของการอ่าน หรือ เขียนอุปกรณ์ อินพุท/เอาท์พุท มีข้อสังเกตุอย่างหนึ่งคือ หลัง T2 ซีพียูจะใส่คาบการรอคอย (Tw) เข้าอัตโนมัติ 1 คาบ ที่เป็นเช่นนี้เพราะว่าในช่วงการทำงาน อินพุท/เอาท์พุท นี้ ระยะเวลาที่สัญญาณ $\overline{IORQ}$ เริ่มเกิดขึ้นจนถึงเวลาที่ซีพียูจะต้องตรวจสอบสัญญาณ การรอคอยนั้นมันสั้นมากหากไม่ใช้ Tw เข้าไปเป็นกรณีพิเศษแล้ว ช่วงเวลานี้อาจจะสั้นเกินกว่าที่ อินพุท/เอาท์พุท พอร์ตใด ๆ จะถอดรหัสของตัวเอง แล้งส่งสัญญาณ $\overline{WAIT}$ ได้ทัน นอกจากนี้หากไม่มีการใส่ TW นี้เข้าไป ยังเป็นการยากมากที่จะออกแบบอุปกรณ์ อินพุท/เอาท์พุท ที่เป็นพวก MOS ให้ทำงานพอดีกับการทำงานของซีพียู

<u>ไซเกิลการขอใช้บัสและตอบสนองการใช้บัส</u>

ที่เวลาใดๆ ของการทำงานของซีพียู อุปกรณ์ภายนอกสามารถส่งสัญญาณมาควบคุมบัส ตำแหน่ง A-15 บัสข้อมูล D7-DO และสัญญาณควบคุม $\overline{MREQ}$ $\overline{RD}$ $\overline{WR}$ $\overline{IORQ}$ และ $\overline{RFSH}$ ได้โดยการส่งสัญญาณ $\overline{BUSRQ}$ เข้ามาอยู่ที่ซีพียู เหตุผลที่ทำเช่นนี้ เนื่องจากเพื่อให้อุปกรณ์ภายนอกสามารถควบคุมการติดต่อโดยตรงระหว่าง หน่วยความจำภายนอกและอุปกรณ์ อินพุท/เอาท์พุท ซึ่งทำงานด้วยความเร็วสูง โดยไม่ต้องมีซีพียู วิธีการแบบนี้เรียกว่า Direct Memory Access (DMA) สัญญาณ $\overline{BUSRQ}$ นี้สามารถถูกตรวจสอบโดยซีพียูที่ขอบพัลส์ขาขึ้นที่คาบเวลาสุดท้ายของแมชชีนไซเกิล ใด ๆ เมื่อพบว่าที่ขา $\overline{BUSRQ}$ นี้มีระดับ "0" แสดงว่ามีการขอใช้บัสภายนอก ดังรูปที่ 22 เมื่อซีพียูได้รับสัญญาณ $\overline{BUSRQ}$ มันจะส่งสัญญาณตอบรับ $\overline{BUSAK}$ เมื่อถึงขอบพัลส์ขาขึ้นของคาบเวลาต่อมา และในขณะเดียวกันบัสตำแหน่งบัสข้อมูลรวมทั้งสัญญาณควบคุมที่ $\overline{MREQ}$ $\overline{RD}$ $\overline{WR}$ $\overline{IORQ}$ และ $\overline{RFSH}$ มีสภาวะเป็นอิมพีแดนซ์สูง ดังนั้นการเปลี่ยนแปลงใด ๆ บนบัสขณะนี้จะไม่มีผลต่อซีพียู และบัสต่าง ๆ ก็จะถูกควบคุมโดยตัวควบคุม DMA เพื่อส่งถ่ายข้อมูล ระหว่างหน่วยความจำ และอุปกรณ์ อินพุท/เอาท์พุท บัสต่าง ๆ จะอยู่ในสภาวะอิมพีแดนซ์สูงตราบเท่าที่ขา $\overline{BUSRQ}$ นี้แอคตีฟอยู่การ ยกเลิกการขอใช้บัสเมื่อเกินสัญญาณ $\overline{BUSRQ}$ นี้ไม่แอคตีฟซึ่ง ซีพียู ตรวจ

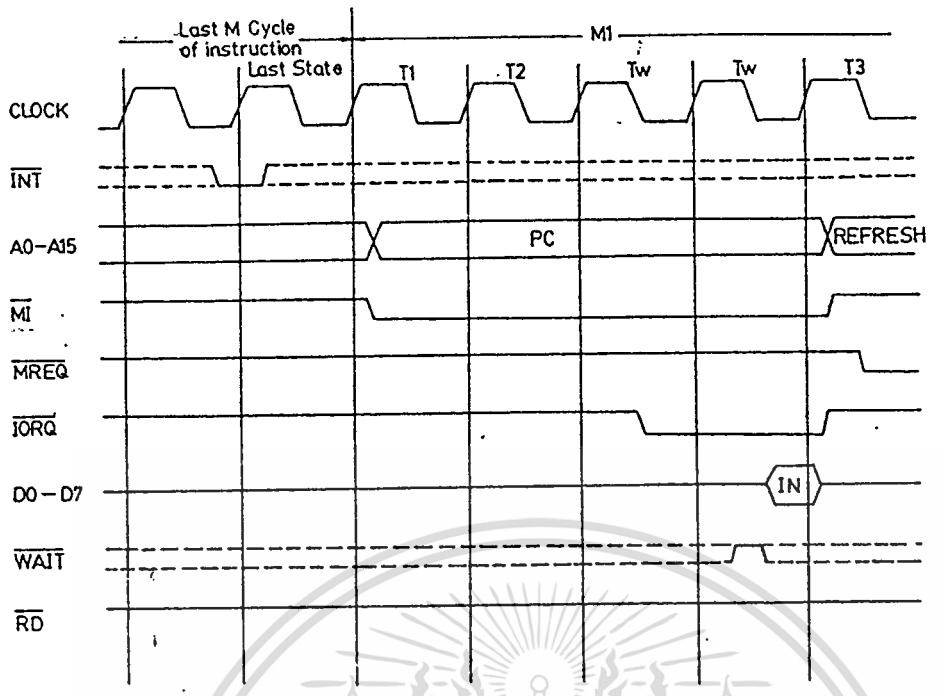


รูปที่ 22 ไซเกิลการใช้ขอบัส

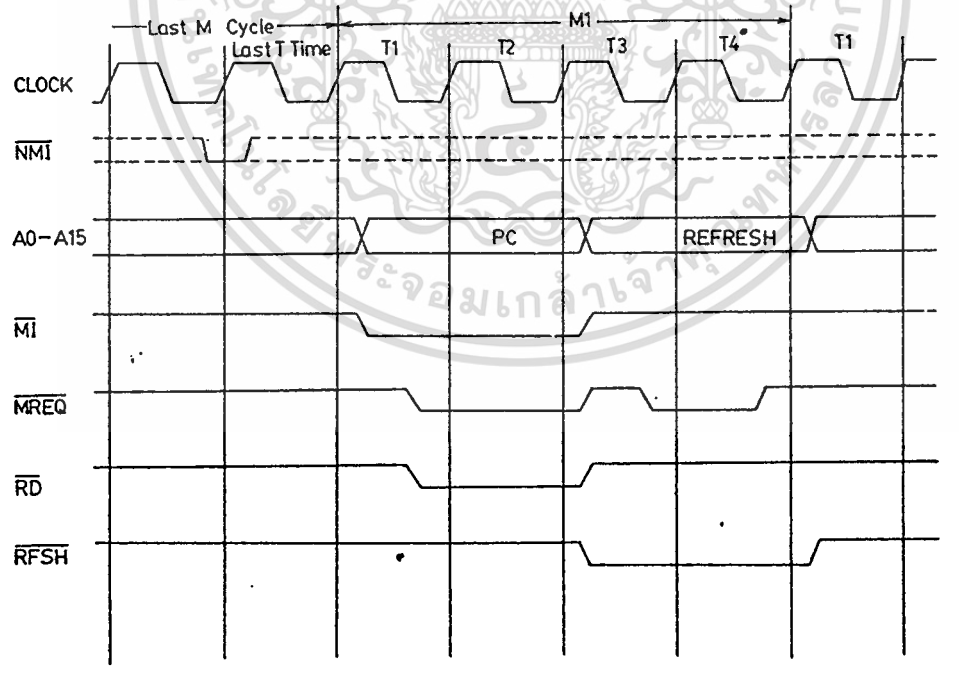
ตรวจสอบที่ ขอบพัลส์ขา ขึ้นของคาบเวลาต่อไปหลังจากที่มีการขอใช้บัสแล้ว จากคาบเวลาต่อไปสัญญาณ $\overline{BUSAK}$ จะหายไปและ ขอบพัลส์ขาขึ้น ของคาบเวลาต่อไป บัสต่าง ๆ ก็กลับสู่สภาพเดิม ข้อสังเกตในการทำ DMA อย่างนี้คือ ถ้าระบบใช้หน่วยความจำแบบไดนามิกแรม และขณะที่อยู่ในและช่วงเวลาการ DMA จะต้องสร้างสัญญาณ รีเฟรช หน่วยความจำแบบไดนามิกแรมนี้ และในขณะที่อยู่ในไซเกิลของ $\overline{BUSRQ}$ ซีพียูจะไม่รับการอินเตอร์รัพต์ทั้งแบบ $\overline{NMI}$ และ $\overline{INT}$

<u>ไซเกิลการขออินเตอร์รัพต์และการตอบรับการอินเตอร์รัพต์</u>

ไซเกิลนี้เป็นอินเตอร์รัพต์แบบ มาสเคเบิล ที่อาจเกิดขึ้นที่ขา $\overline{INT}$ ถ้าอินเตอร์รัพต์ ฟลิปฟลอบ (IFF) ถูกเซตให้ยอมรับการอินเตอร์รัพต์ และในขณะนั้นไม่มีการขอใช้บัสโดยสัญญาณ $\overline{BUSRQ}$ แล้ว แสดงว่าซีพียูสามารถที่จะรับสัญญาณ $\overline{INT}$ จากอุปกรณ์ภายนอกได้สัญญาณการขอ อินเตอร์รัพต์ จากอุปกรณ์ภายนอกที่ขอบพัลส์ขานี้ ซีพียูจะทำการตรวจสอบที่ขอบพัลส์ขาขึ้นของคาบเวลาสุดท้าย ของแมชชีนไซเกิลการขอ และการตอบรับการอินเตอร์รัพต์แบบ มาสเคเบิล การกระทำของ ซีพียูในไซเกิลของการขอ และตอบรับการ อินเตอร์รัพต์แบบนอนมาสเคเบิล นี้ แสดงดังที่รูปที่ 23



รูปที่ 23 ไซเกิลการรับและการตอบการอินเตอร์รัพต์

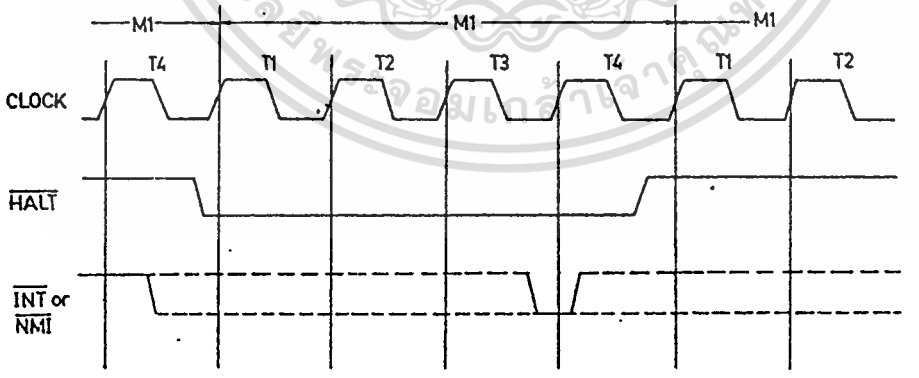


รูปที่ 24 ไซเกิลการรับและตอบรับอินเตอร์รัพต์แบบนอนมาสเคเบิล

สัญญาณการขออินเตอร์รัพต์ แบบนี้จะถูกส่งมาจากอุปกรณ์ภายนอกเข้ามาทางขา $\overline{NMI}$ ของซีพียูและซีพียูไม่สามารถที่จะกันไม่ให้รับ การอินเตอร์รัพต์แบบนี้ได้ เนื่องจากสัญญาณที่ขานี้จัดให้มีลำดับความสำคัญสูงกว่าสัญญาณที่ขา $\overline{INT}$ สัญญาณ $\overline{NMI}$ นี้จะแอคตีฟที่ขอบพัลส์ขาลงแต่ซีพียูจะทำการตรวจสอบ สัญญาณที่ขานี้ในเวลาเดียวกับการตรวจสอบที่ขา $\overline{INT}$ ถ้าขา $\overline{NMI}$ มีระดับ 0 แสดงว่ามีการขออินเตอร์รัพต์แบบ นอนมาสเคเบิล และซีพียูจะทำการตอบสนองในลักษณะเดียวกับการอ่านข้อมูล ข้อแตกต่างมีเพียงแต่ ซีพียู จะไม่สนในข้อมูลจากบัสข้อมูล แต่จะส่งค่าโปรแกรมเคาเตอร์ไปเก็บไว้ในสแตคภายนอก และกระโดดไปทำงานที่ตำแหน่ง 0066H โดยอัตโนมัติดังนั้นโปรแกรมปริการอินเตอร์รัพต์แบบนี้จะต้องเริ่มต้นที่ตำแหน่ง 0066H นี้เท่านั้น

ไซเกิลการออกคำสั่ง HALT

การกระทำโปรแกรมเมื่อใดก็ตามที่ซีพียูปฏิบัติคำสั่ง$\overline{HALT}$ สัญญาณ $\overline{HALT}$ จะแอคตีฟและจากนั้นซีพียูจะสร้างไซเกิล M1 และปฏิบัติคำสั่งในลักษณะ NOP แต่ไม่มีการเพิ่มค่าโปรแกรมเคาน์เตอร์การปฏิบัติคำสั่ง NOP นี้จะเกิดเรื่อยไปจนกระทั่ง ซีพียูได้รับสัญญาณอินเตอร์รัพต์ ที่ขา $\overline{RESET}$ $\overline{NMI}$ หรือ $\overline{INT}$ (เมื่อ IFF=1)สัญญาณอินเตอร์รัพต์ $\overline{NMI}$ และ $\overline{INT}$ จะถูกตรวจสอบที่ขอบพัลส์ขึ้นของ T4 ดังที่รูป 25 ถ้าซีพียูได้รับสัญญาณการอินเตอร์รัพต์ มันจะะออกจากสภาวะของการ $\overline{HALT}$ ที่ขอบพัลส์ขาขึ้นของคาบเวลาต่อไป จากนั้นซีพียูจะตอบรับและกระทำการบริการการขออินเตอร์รัพต์ตามชนิดของการอินเตอร์รัพต์ ที่เกิดขึ้น



รูปที่ 25 ไซเกิลการออกคำสั่ง $\overline{HALT}$

จุดประสงค์ของการทำคำสั่ง NOP ในช่วง $\overline{HALT}$ ก็เพื่อที่จะให้สัญญาณรีเฟรชยังคงทำ

งานต่อไป ในแต่ละไซเกิล $\overline{HALT}$ ซึ่งก็คือ ไซเกิล M1 ธรรมดา ยกเว้นแต่ว่ามันจะไม่สนใจข้อมูลจากหน่วยความจำและรหัสคำสั่ง NOP จะเกิดขึ้นภายในเอง

บทที่ 2

การเชื่อมต่อไมโครโปรเซสเซอร์กับหน่วยความจำ

ไมโครโปรเซสเซอร์ไม่สามารถทำงานอิสระได้ แต่จะต้องผสมกับหน่วยต่าง ๆอีกคือ หน่วยความจำและอินพุท/เอาท์พุทจึงจะสามารถทำงานเพื่อให้เป็นระบบไมโครคอมพิวเตอร์ได้ ซึ่งหน้าที่ที่สำคัญของหน่วยความจำในระบบไมโครคอมพิวเตอร์ก็คือ

1. เป็นหน่วยที่ใช้เก็บลำดับขั้นของคำสั่ง หรือที่เรียกว่าโปรแกรม เพื่อกำหนดให้ไมโครโปรเซสเซอร์ทำงานตามที่ต้องการ

2. เป็นหน่วยที่ใช้เก็บข้อมูลชั่วคราว ซึ่งหมายถึงสามารถเขียนข้อมูลเข้าไปในหน่วยความจำหรือนำข้อมูลเก่ามาจากหน่วยความจำ

หน่วยความจำ

หน่วยความจำในระบบคอมพิวเตอร์ทั้งหลาย คือ อุปกรณ์อิเลกทรอนิกส์ที่ใช้ในการจดจำข้อมูลที่หน่วยความจำนี้ได้ จะอยู่ในรูปสัญญาณแม่เหล็ก หรือสัญญาณไฟฟ้าที่สามารถแสดงได้ว่าข้อมูลที่มีอยู่นั้นมีสภาวะ 0 หรือ 1 ในยุคก่อนหน่วยความจำ ในระบบคอมพิวเตอร์ส่วนใหญ่จะเป็นสารแม่เหล็ก (Magnetic core Memory) คือใช้แกนเฟอร์ไรท์และใช้ทิศทางของสนามแม่เหล็กเป็นตัวบอกให้ทราบว่าขณะนั้นเป็นสภาวะ 0 หรือ 1 แต่ในปัจจุบันหน่วยความจำได้วิวัฒนาการมาใช้อุปกรณ์ สารกึ่งตัวนำ (Semiconductor Memory) ซึ่งมีคุณสมบัติที่เหนือกว่าในหลาย ๆ ด้าน ดังนั้นในที่นี้จะกล่าวถึงแต่หน่วยความจำ แบบสารกึ่งตัวนำเท่านั้น ลักษณะของหน่วยความจำแบบนี้คือ

1. Nondestructive ewadout คือเมื่อทำการอ่านข้อมูลจากหน่ายความจำแล้วข้อมูลที่ตำแหน่งนั้นจะไม่เปลี่ยนแปลง (หน่วยความจำแบบ Magnetic core ข้อมูลจะมีการเปลี่ยนแปลงหลังการอ่าน) ดังนั้นจึงไม่จำเป็นต้องทำการเขียนข้อมูลกลับเข้าไปอีก

2. Volotile คือความต้องการไฟเลี้ยง หน่วยความจำที่สามารถอ่านหรือเขียนได้ เช่นหน่วยความจำแบบ แรม ข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยงหน่วยความจำ แต่หน่วยความจำประเภทอ่านได้อย่างเดียว เช่น รอม เป็นหน่วยความจำที่ข้อมูลจะไม่มีการเปลี่ยนแปลงแม้ว่าจะไม่มีไฟเลี้ยงก็ตาม ดังหน่วยความจำแบบรอม จะเป็นหน่วยความจำ ชนิด Nonvolatile

3. Single chip from หน่วยความจำแบบสารกึ่งตัวนำนี้จะบรรจุอยู่ในชิพไอซี ซึ่งจะมีขนาด ต่าง ๆ กัน เช่นขนาดของคำ (word lengh) จำนวนตำแหน่งของหน่วยความจำและโครงสร้างภายใน ส่วนการถอดรหัสตำแหน่งและ การเข้าถึงข้อมูลจะเป็นส่วนที่อยู่ภายในไอซีเอง

ชนิดของหน่วยความจำ

หน่วยความจำอาจแบ่งตามลักษณะการใช้ และการอ่านเขียนข้อมูลได้ 3 แบบใหญ่ๆ คือ

1. Read Only Memory (ROM)
2. Random Access Memory (RAM)
3. Serial Access Memory

ซึ่งลักษณะของหน่วยความจำทั้ง 3 แบบอาจกล่าวโดยสรุปได้ดังนี้ คือ Read only Memory (ROM) เป็นหน่วยความจำที่ใช้เก็บข้อมูลแบบถาวร หรือกึ่งถาวร คือข้อมูลที่จะต้องถูกเขียนลงไปใน หน่วยความจำตั้งแต่ต้น หลังจากนั้นก็ใช้เรียกใช้เพียงอย่างเดียวแต่ไม่สามารถแก้ไขเปลี่ยนแปลงได้อีก ดังนั้น เราเรียกหน่วยความจำแบบนี้ว่า Read only หน่วยความจำแบบรอมนี้เป็นหน่วยความจำแบบ Nonvolatile คือข้อมูลในรอมจะไม่สูญหายเมื่อไม่มีไฟเลี้ยง หน่วยความจำรอม จะใช้ระบบไมโครคอมพิวเตอร์ทุกเครื่อง เพื่อเก็บโปรแกรมสำหรับใช้ระบบเริ่มต้นทำงานที่เรียกว่า โปรแกรมโมนิเตอร์หรือ โปรแกรม Bootstrap รอมสามารถจำแนกเป็นชนิดต่าง ๆ ได้ตามลักษณะการเขียนข้อมูลเข้าไปในหน่วยความจำดังนี้

Mask Programmed ROM เป็นรอม ชนิดที่ได้ทำการเขียนข้อมูลเข้าไปในหน่วยความจำตั้งแต่ตอนสร้างชิฟ และไม่สามารถแก้ไขข้อมูลภายในอีก รอมแบบนี้จะต้องสร้างจำนวนครั้งละมาก ๆ เนื่องจากต้นทุนการผลิตค่อนข้างสูง

Programmable Read Only Memory หรือ PROM หน่วยความจำแบบนี้ผู้ใช้สามารถโปรแกรมเองได้ โดยใช้กรรมวิธีจ่ายพัลส์แรงดันสูงเข้าไปทำลายฟิวส์ ภายในตัวไอซีหน่วยความจำ เพื่อทำให้เป็นลอจิก 0 หรือ 1 ตำแหน่งที่กำหนดและเมื่อโปรแกรมเข้าไปแล้วไม่สามารถทำการแก้ไขได้เช่นกัน

Eraable Programmable Read Only Memory หรือ EPROM หน่วยความจำชนิดนี้ ผู้ใช้สามารถโปรแกรมข้อมูลลงไปให้และสามารถลบได้โดยใช้รังสี Ultra violet ฉายผ่านช่องกระจกตัว ไอซี และหลังจากที่ทำการลบข้อมูลออกแล้ว ก็สามารถโปรแกรมข้อมูลใหม่ลงไปได้อีก

Electrically Aoterable Read Only Memory หรือ EAROM เป็นหน่วยความจำที่สามารถโปรแกรมข้อมูลเข้าไปได้ และสามารถลบออกได้โดยใช้สัญญาณไฟฟ้าทำให้การโปรแกรมและการลบทำได้โดยสะดวก และอาจทำได้โดยไม่ต้องถอดออกจากวงจร

RANDOM ACCESS MEMORY หน่วยความจำแบบแรมนี้ เป็นหน่วยความจำที่มี

การทำงานที่ต่างจากรอม คือมันสามารถที่จะทำการเขียนหรือ อ่านข้อมูลแต่ละคำหรือแต่ละบิต ณ ที่ตำแหน่งใด ๆ ในพื้นที่ของหน่วยความจำนี้จะใช้เวลาเท่า ๆ กันในกรณีของรอการเขียนข้อมูลเข้าหน่วยความจำ จะใช้เวลามากกว่าการอ่านข้อมูลมากหน่วยความจำแบบ แรมจะเป็นชนิด Volatile คือต้องการไฟเลี้ยงตลอดเวลาเมื่อขาดไฟเลี้ยงข้อมูลในแรมจะหายไปแรมจะแบ่งออกได้ 2 แบบ คือ

Static RAM หรือเรียกย่อ ๆ ว่า SRAM แรมชนิดนี้ หน่วยความจำแต่ละเซลจะใช้ลักษณะของวงจร ฟลิปฟลอป เป็นพื้นฐาน ดังนั้นเมื่อไม่มีการเขียนข้อมูลเข้าไปใหม่ ข้อมูลนั้น ๆ จะคงที่ตลอดไปเท่าที่มีไฟเลี้ยงอยู่

Dynamic RAM หรือ DRAM ลักษณะของแรมชนิดนี้ ใช้การเก็บประจุที่ขาเกตของ MOSFET เพื่อเป็นการเก็บข้อมูล และเมื่อมีการอ่านข้อมูลออกมา ประจุที่ถูกเก็บไว้จะคายออกมาหมดไป ซึ่งถ้าเป็นเช่นนี้ หน่วยความจำก็จะเป็นแบบ Destructive ดังนั้น DRAM คือ มีความหนาแน่นของเซลหน่วยความจำสูงกว่า มีความสิ้นเปลืองกำลังต่ำกว่าและใช้เวลาในการเข้าข้อมูลเร็วกว่าแต่มีข้อเสียในเรื่องการที่จะต้องทำการ รีเฟรช ซึ่งจะยุ่งยากในการใช้มากกว่า SRAM

SERIAL ACCESS MEMORY หน่วยความจำแบบนี้ใช้วิธีการ เก็บข้อมูลตามลำดับก่อนหลังตามกันไป ดังนั้นเวลาอ่านจะต้องเรียงลำดับด้วย เช่นเดียวกับเทป ซึ่งลักษณะของหน่วยความจำแบบนี้คล้ายกับวงจรชีฟรีจีสเตอร์ หน่วยความจำแบบนี้ Volatile และ Nonvolatile เช่น

Magmetic Bubble Memory หน่วยความจำแบบนี้มีความจุข้อมูลสูงใกล้เคียงกับเทป สามารถเขียนและอ่านข้อมูลได้เช่นเดียวกันกับ เทป และหน่วยความจำชนิด Nonvola tile ไม่จำเป็นต้องมีการรีเฟรช แต่ความเร็วในการอ่านหรือ เขียนข้อมูลค่อนข้างช้ากว่าแรมแบบสารกึ่งตัวนำแต่มีความเร็วสูงกว่าเทปมาก หน่วยความจำแบบนี้มีแนวโน้มจะเข้ามาแทนที่ Disk ในอนาคต

Charge Coupled Device (CCD) หน่วยความจำแบบนี้สามารถเก็บข้อมูลได้สูง สามารถอ่านหรือเขียนได้ แต่เป็นแบบ Volatile ความเร็วการอ่านและเขียนข้อมูลค่อนข้างสูงเกือบเท่ากัน แรม แบบสารกึ่งตัวนำ CCD มีราคาถูก แต่การใช้งานยุ่งยากจึงไม่เป็นที่นิยม

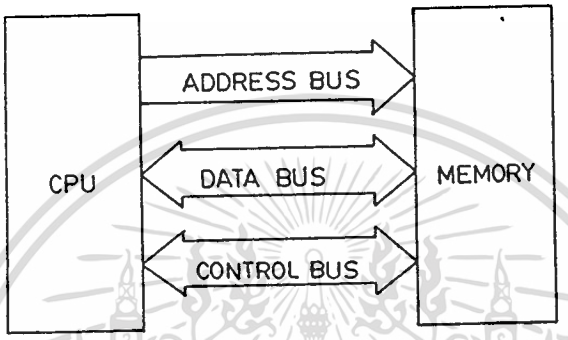
การต่อหน่วยความจำกับไมโครโปรเซสเซอร์

ในการเชื่อมกันระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำโดยทั่วไปนั้น จะต้องมีสัญญาณการเชื่อมต่อไปนี้

1. บัสตำแหน่ง ซีพียูจะใช้ข้อมูลแบบบัสนี้สำหรับการเลือกตำแหน่งของหน่วยความจำที่จะต้องใช้เป็นต้นทางข้อมูล (เมื่อทำการอ่าน) หรือตำแหน่งปลายทางข้อมูล (เมื่อทำการเขียน)

2. บัสข้อมูล ใช้สำหรับเป็นทางผ่านของข้อมูล ซีพียู เพื่อออกไปยังหน่วยความจำ หรือข้อมูลจากหน่วยความจำเพื่อมายัง ซีพียู

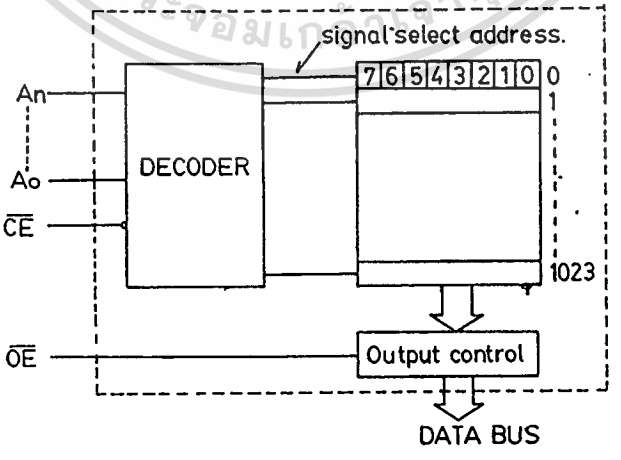
3. บัสควบคุม บัสนี้ใช้สำหรับการส่งสัญญาณควบคุมการทำงานในการอ่านหรือการเขียนข้อมูล เช่นสัญญาณ $\overline{RD}$ $\overline{WR}$ และสัญญาณเพื่อให้มีการทำงานพร้อมกันเป็นต้น ซึ่งลักษณะของการเชื่อมต่อซีพียูกับหน่วยความจำ แสดงได้ดังรูปที่ 26



รูปที่ 26 การเชื่อมต่อหน่วยความจำกับซีพียู

<u>ลักษณะภายในหน่วยความจำแบบ ROM</u>

ลักษณะการจัดโครงสร้างภายในของหน่วนความจำรอมแบบที่เป็น EPROM แสดงได้ดัง



รูปที่ 27 การจัดโครงสร้างของหน่วยความจำแบบ EPROM

รูปที่ 27 ซึ่งหน่วยความจำแบบนี้สามารถโปรแกรมได้ และอ่านได้ ในการกำหนดขนาดของหน่วยความจำแบบ EPROM สามารถกำหนดได้ดังนี้ เช่น 1024 x 8 หมายถึงหน่วยความจำที่มี 1024 ตำแหน่ง หรือ 1K และแต่ละตำแหน่งประกอบด้วย 8 บิต ถ้าหน่วยความจำมี 1024 ตำแหน่ง ก็จะต้องมีสัญญาณในการกำหนดตำแหน่งเท่ากับ 10 สัญญาณ นั่นคือ 2^{10} = 1024 สัญญาณเหล่านี้ปกติจะมาจากบัสตำแหน่งของไมโครโปรเซสเซอร์ ไอซีหน่วยความจำที่ใช้กันอยู่หลายขนาด เช่น ถ้าเป็น EPROM เบอร์ 2716 จะมีขนาด 2K x 8 เบอร์2732 มีขนาด 4K x 8 เบอร์ 2764 มีขนาด 8K x 8 เป็นต้น

<u>ลำดับการอ่านข้อมูลจากหน่วยความจำ EPROM</u>

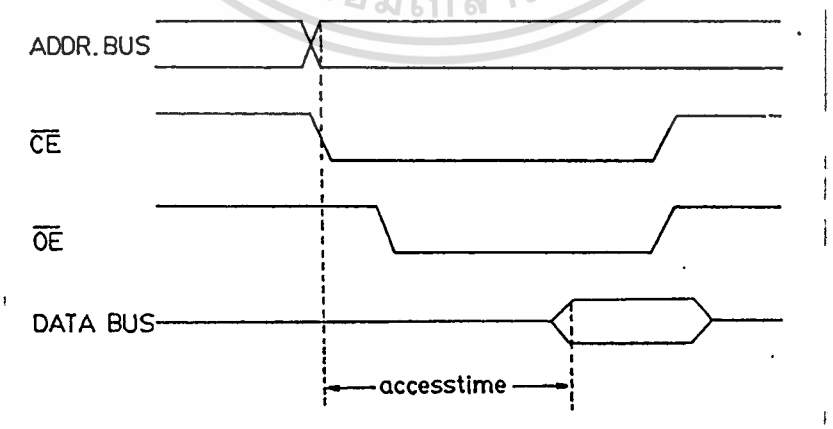
ในการใช้ซีพียูจะอ่านข้อมูลจากหน่วยความจำแบบ EPROM จะมีลำดับดังนี้

1. ซีพียูส่งข้อมูลของตำแหน่งที่ต้องการอ่านออกไปบนบัสตำแหน่ง

2.ซีพียูกำเนิดสัญญาณเลือกใช้ชิพ (Chip enable และ Output enable) และส่งไปยังหน่วยความจำ ซึ่งสัญญาณนี้อาจส่งไปในเวลาเดียวกันกับการส่งสัญญาณใน การกำหนดตำแหน่งก็ได้

3. หลังจากนั้น ซีพียูต้องรอสักระยะเวลาหนึ่ง เพื่อให้หน่วยความจำทำการอ่านข้อมูลเพื่อส่งออกไปยังบัสข้อมูลของระบบซึ่งช่วงเวลานี้เรียกว่า Access time ซึ่ง มีค่าอยู่ระหว่าง 150-450 nS และหลังจากนั้น ซีพียูจึงทำการอ่านข้อมูลในช่วงนี้เข้าไป

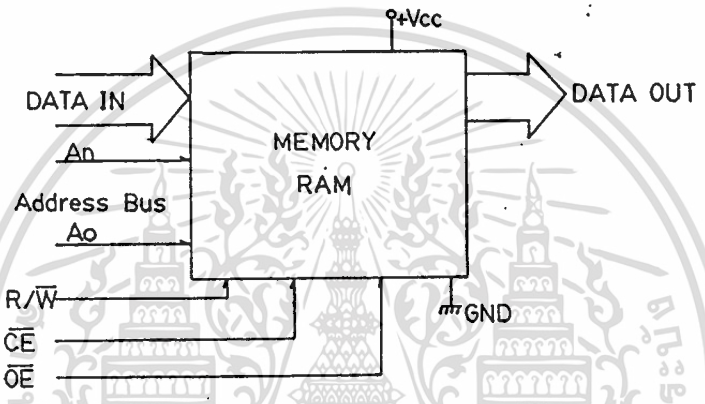
4. สัญญาณเลือกชิพเลิกแอคตีฟ บัสข้อมูลกลับสู่สภาวะอิมพีแดนซ์สูงซึ่งลำดับขั้นการอ่าน EPROM นี้ สามารถเขียนเป็นไดอะแกรมเวลาแสดงได้ดังรูป 28



รูปที่ 28 ไดอะแกรมเวลาของการอ่านข้อมูลจาก EPROM

ลักษณะหน่วยความจำแบบ RAM

หน่วยความจำแบบแรมก็จะมีลักษณะภายนอก คล้าย ๆ กับหน่วยความจำแบบรอมแต่แตกต่างกันที่แรม จะมีสายสัญญาณเพิ่มขึ้นมาอีกเส้นหนึ่ง คือสัญญาณควบคุมการอ่านและเขียน บล็อกไดอะแกรมของแรมแสดงได้ดังรูปที่ 29 ซึ่งในที่นี้จะใช้แรมแบบสแตติกบล็อกไดอะแกรม บัสสำหรับข้อมูลเข้าและส่งข้อมูลออกจะเป็นบัสเดียวกันแต่เป็นบัสแบบ 2 ทิศทางเป็นแบบบัสขาเข้าเมื่ออยู่ในช่วงของการเขียน และเป็นแบบบัสขาออกเมื่ออยู่ในช่วงของการอ่านหน่วยความจำแรมแบบ สแตติก หรือ SRAM มีหลายขนาดให้เลือกใช้เช่น เบอร์ 2114 เป็นแรมขนาด 1K x 4 เบอร์ 6116 เป็นแรมขนาด 2K x 8 และเบอร์ 6264 เป็นแรมขนาด 8K x 8 เป็นต้น



รูปที่ 29 บล็อกไดอะแกรมของหน่วยความจำแบบแรม

ลำดับขั้นในการอ่านข้อมูลแบบแรม

ในการอ่านข้อมูลจากแรมนั้น ต้องใช้สัญญาณควบคุมการอ่าน และเขียนมาร่วมในการควบคุมการทำงานด้วยโดยการกำหนดให้สัญญาณควบคุมการอ่านและเขียนนี้ แอคตีฟตามการลักษณะของ การอ่านหรือเขียนข้อมูลดังต้องการ เช่น ถ้าให้สัญญาณนี้อยู่ในสภาวะแอคตีฟอ่าน (สมมุติ = 1) การทำงานก็เป็นการอ่านข้อมูลจากหน่วยความจำตามตำแหน่งที่ระบุ ซึ่งลำดับขั้นของการเขียนไดอะแกรมเวลาได้ดังรูปที่ 30 และมีลำดับดังนี้

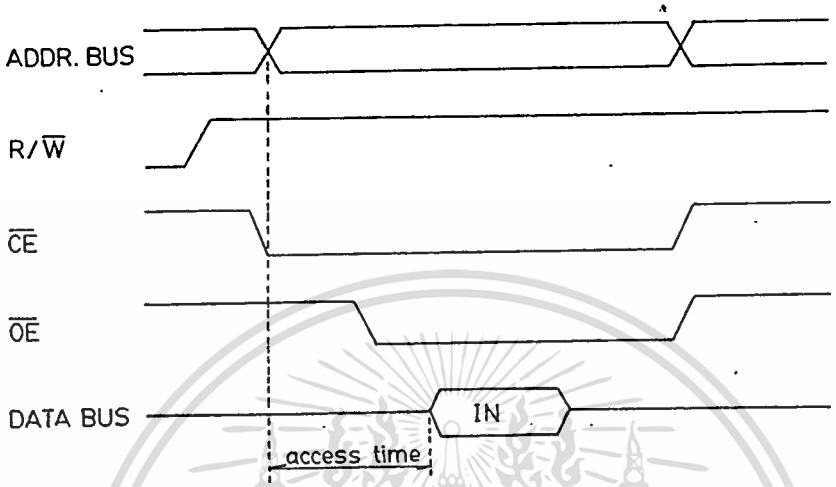
1. ซีพียูส่งข้อมูลของตำแหน่งที่ต้องการอ่านออกไปบนบัสตำแหน่ง (กำหนดให้ในขณะนี้สัญญาณการอ่านแอคตีฟ)

2. ซีพียูส่งสัญญาณเลือกใช้ชิพและส่งไปยังหน่วยความจำ

3. ซีพียูรอเวลาสักระยะหนึ่ง นับตั้งแต่ส่งข้อมูลเพื่อกำหนดตำแหน่งเข้าไปซึ่งเวลานี้ก็คือ Access time ซึ่งอาจมีค่าประมาณ 35-200 nS แล้วแต่ชนิดและขนาดของแรมหลังจากช่วง

Access time ผ่านไปแล้ว ข้อมูลจากหน่วยความจำก็จะปรากฏบนบัสข้อมูล ซึ่งซีพียูจะอ่านข้อมูลในช่วงนี้เข้าไป

4. สัญญาณเลือกชิพเลิกแอคตีฟ บัสข้อมูลกลับสู่สภาวะอิมพีแดนซ์สูง



รูปที่ 30 ไดอะแกรมเวลาของการอ่านข้อมูลจากแรม

<u>ลำดับขั้นการเขียนข้อมูลลงหน่วยความจำแบบแรม</u>

ในการเขียนข้อมูลลงหน่วยความจำแบบแรม จะมีลำดับขั้นดังนี้

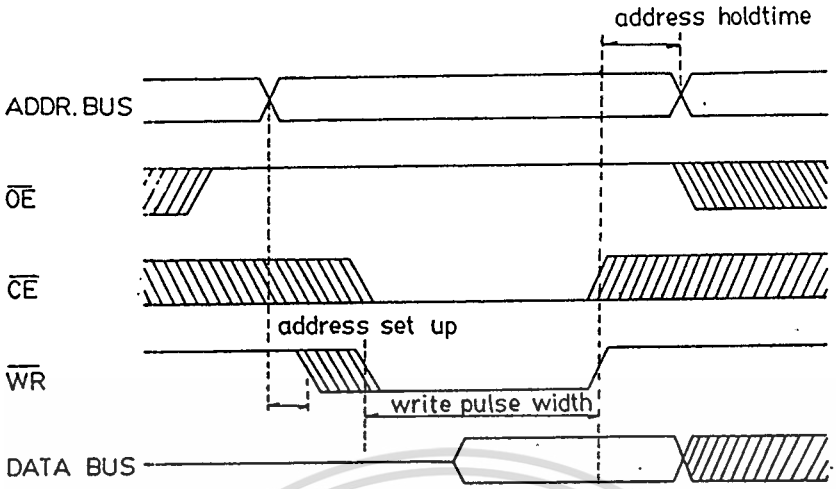
1. ซีพียู ส่งข้อมูลของตำแหน่งที่ต้องการให้เขียนไปบนบัสตำแหน่ง และอาจพร้อมทั้งสัญญาณการเลือกใช้ชิพ ($\overline{CE}$)

2. ข้อมูลที่จะเป็นข้อมูลอินพุท ถูกส่งเข้ามายังอินพุทของแรม

3. หน่วยความจำต้องการเวลาสักระยะหนึ่ง เพื่อให้ข้อมูลในการกำหนดตำแหน่งอยู่ในสภาวะคงที่ซึ่งช่วงเวลานี้คือ Address setup time ซึ่งจะมีค่าประมาณ 20 nS จากนั้นระบบจึงส่งสัญญาณควบคุมการเขียนให้ข้อมูลอยู่ในสภาวะแอคตีฟ

4. หลังจากที่สัญญาณควบคุมการเขียนแอคตีฟสักระยะหนึ่ง หน่วยความจำจะทำการนำข้อมูลจากอินพุท เข้าเขียนเข้าไปยังหน่วยจำ และเมื่อสัญญาณควบคุมการเขียนเลิกแอคตีฟก็เป็นอันสิ้นสุดขบวนการเขียนลำดับขั้นของการเขียนข้อมูลลงแรมสามารถแสดงได้โดยไดอะแกรมเวลาดังรูปที่ 31

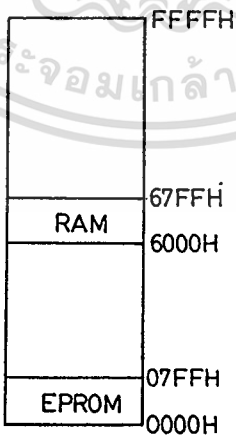
จากรูปเห็นได้ว่ากำหนดให้สัญญาณ $\overline{OE}$ อยู่ในสภาวะไม่แอคตีฟ เพื่อไม่ให้มีสัญญาณเอาท์พุทออกมาที่บัสข้อมูล แต่ถ้าสัญญาณ $\overline{OE}$ อยู่ในสภาวะแอคตีฟ และขณะนั้น สัญญาณ $\overline{CE}$ แอคตีฟด้วยก็จะมีสภาวะเป็นอิมพีแดนซ์สูงเช่นกัน



รูปที่ 31 ไดอะแกรมเวลาของการเขียนหน่วยความจำ

<u>การต่อหน่วยความจำกับซีพียู Z-80</u>

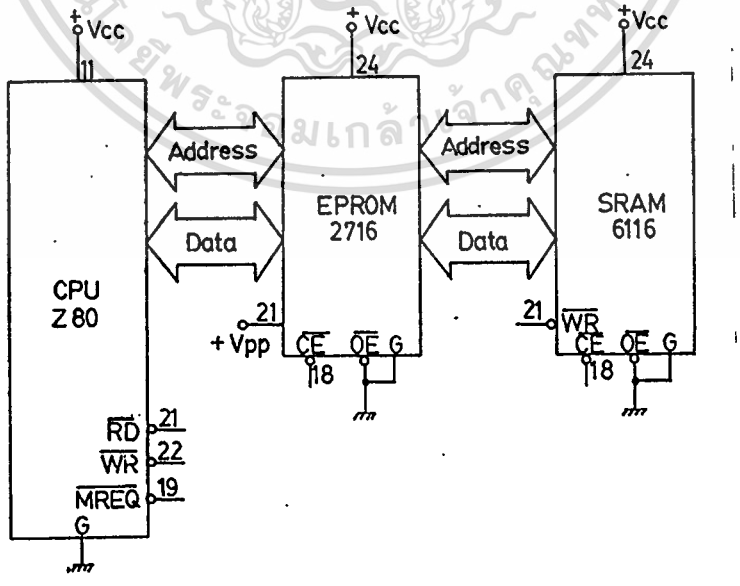
ซีพียู Z-80 มีสัญญาณในการเลือกตำแหน่ง 16 สัญญาณ คือ A15-A0 นั่นหมายความว่าซีพียูสามารถติดต่อกับหน่วยความจำได้ถึง 2^{16} หรือ 65536 ตำแหน่งดังนั้นในการออกแบบ



รูปที่ 32 แสดงการจัดแบ่งหน่วยความจำ หรือ Memory mep

ระบบไมโครคอมพิวเตอร์เราต้องทำการจัดสรรเนื้อที่เสียก่อนว่าเราจะใช้หน่วยความจำส่วนไหนเพื่อประโยชน์อะไร และต้องใช้หน่วยความจำแบบใด ดังนั้นผู้ออกแบบระบบ ต้องทำการสร้างตารางเพื่อจัดสรรหน่วยความจำหรือที่ เรียกว่า Memory Map ขึ้นเพื่อกำหนดตำแหน่งของการที่จะใช้หน่วยความจำแบบรอมหรือแบบแรมซึ่งตัวอย่างของ Memory map แสดงได้ดังรูปที่ 32

จากรูปจะเห็นได้ว่าตำแหน่งต่ำสุดของ Memory map คือตำแหน่งที่ 0000H จะต้องใช้หน่วยความจำแบบรอม เนื่องจากเมื่อทำการรีเซตซีพียู ตำแหน่งเริ่มต้นที่ซีพียูจะทำการอ่านในครั้งแรกคือตำแหน่งที่ 0000H นี้เอง และข้อมูลนี้จะต้องเป็นคำสั่งเพื่อให้ซีพียูมีการทำงานได้ต่อไปนั้นคือโปรแกรมในส่วนแรกที่บรรจุในหน่วยความจำแบบรอมนี้ จะเป็นโปรแกรมที่เรียกว่า Bootstrap program หรือมอนิเตอร์โปรแกรมนั่นเอง และส่วนนอกเหนือจากนั้นอาจกำหนดให้ส่วนใดเป็นรอมหรือส่วนใดเป็นแรมก็ได้ จากรูป เราได้ทำการแบ่งหน่วยความจำออกเป็นส่วน ๆหรือ เรียกว่า แบงก์ (Bank) โดยมีส่วนขนาดแบงก์ละ 2k แบงก์ 0 คือ ตั้งแต่ตำแหน่ง 0000H-07FFH กำหนดให้พื้นที่ของ ROM A แบงก์ ตั้งแต่ตำแหน่ง 6000H-67FFH ใช้เป็นพื้นที่ของ RAM A จาก Memory map เห็นได้ว่าแต่ละแบงก์มีขนาด 2K ดังนั้นจะให้หน่วยความจำ EPROM เบอร์ 2716 สำหรับพื้นที่ที่เป็นรอมและใช้ SRAM เบอร์ 6116 สำหรับพื้นที่ๆเป็นแรมซึ่งหน่วยความจำทั้งสองนี้มีขนาด 2K เท่ากัน ดังนั้นหน่วยความจำทั้งสองจะต้องมีอินพุทเพื่อ การกำหนดตำแหน่ง 11 เส้นคือ A10-A0 ซึ่งอินพุททั้ง 11 นี้สามารถต่อได้โดยตรงกับบัสตำแหน่งของซีพียู ที่ A10-A0 ส่วน A15-A11 ของซีพียูจะนำมาใช้เพื่อเลือกว่าจะใส่หน่วยความจำในพื้นที่ส่วนใด วงจรในการต่อหน่วยความจำทั้งสองเข้ากับบัสข้อมูลของซีพียูแสดงได้ดังรูปที่ 33



รูปที่ 33 แสดงการต่อหน่วยความจำเข้ากับบัสตำแหน่งและบัสข้อมูล

บทที่ 3

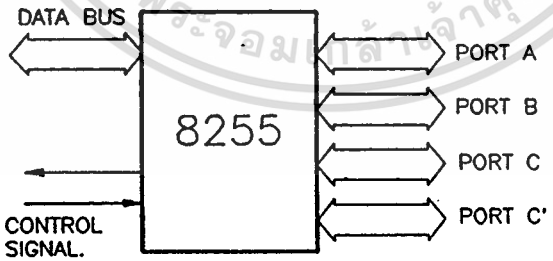
อินพุท/เอาท์พุท พอร์ต

การใช้งานไมโครโปรเซสเซอร์ จะต้องมีการเชื่อมต่อ กับอุปกรณ์ภายนอก เช่น สวิตช์ รีเลย์ หรืออุปกรณ์ตรวจจับ อื่น ๆการเชื่อมต่อดังกล่าวจะต้องเชื่อมต่อกับพอร์ตอินพุท/เอาท์พุทเพื่อให้ไมโครโปรเซสเซอร์ส่งสัญญาณไปยังอุปกรณ์ต่าง ๆ ตามเงื่อนไขที่เกิดขึ้นและสามารถตรวจสอบได้ด้วย ไมโครโปรเซสเซอร์เอง

การเชื่อมต่อที่ง่ายที่สุด คือการเชื่อมต่อโดยการใช้เกตลอจิก 3 สถานะ โดยสัญญาณควบคุมพอร์ตอินพุท จะเป็นตัวเปิดเกตให้ข้อมูลเข้าสู่บัส และไมโครโปรเซสเซอร์อ่านเข้าไป แต่สำหรับพอร์ตเอาท์พุทจะใช้แลตช์ฟลิปฟลอป ทำหน้าที่รับสัญญาณข้อมูลจากไมโครโปรเซสเซอร์ที่ส่งเข้าไปในบัส และได้รับการจับไว้ที่พอร์ตและไมโครโปรเซสเซอร์จะส่งสัญญาณมาควบคุมอีกที

อินพุท/เอาท์พุท ที่ใช้เกตขนาดเล็กดังกล่าว มีจุดอ่อนในเรื่องของจำนวนไอซีบริษัทผู้ออกแบบไมโครโปรเซสเซอร์ส่วนใหญ่จึงออกแบบชิฟ เพื่อทำหน้าที่อินพุท/เอาท์พุท ของระบบ และมีข้อดีในเรื่องการทำงาน สำหรับ ซีพียู Z-80 แล้ว ชิพไอซี ที่รู้จักกันมากที่สุด คือ ไอซี 8255 ของบริษัทอินเทล และสามารถประยุกต์ใช้กับ ซีพียู Z-80 ได้ง่าย

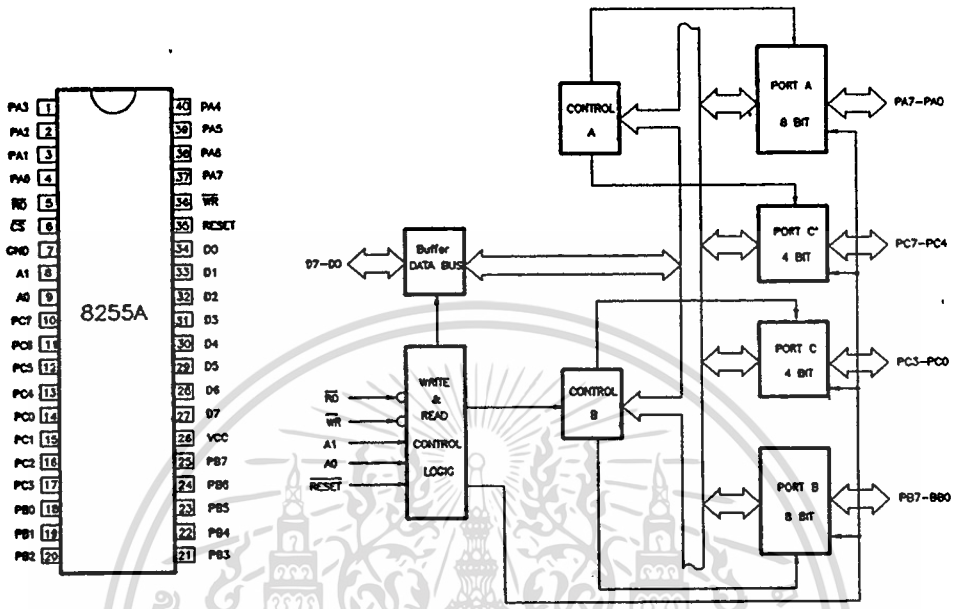
8255 เป็นไอซีที่มี 40 ขา สามารถต่อเป็นพอร์ตให้ไมโครโปรเซสเซอร์ได้ 3 พอร์ต โดยมีโครงสร้างพื้นฐานแสดงได้ดังรูปที่ 34



รูปที่ 34 แผนผังแสดงโครงสร้างของไอซี 8255

การเรียกพอร์ตของ 8255 จะเรียกพอร์ตต่าง ๆ ว่า พอร์ต A พอร์ต B พอร์ต C

โดยพอร์ต C แยกออกเป็น 2 ส่วนคือ พอร์ต C ล่างหรือตั้งแต่ PC_0-PC_3 จำนวน 4 บิต และพอร์ต C บน หรือตั้งแต่ PC_4-PC_7 ที่พิเศษคือพอร์ตทุกพอร์ตใช้ได้ทั้ง อินพุทและเอาท์พุทพอร์ต



รูปที่ 35 แสดงผังภายในและการจัดขาของไอซี 8255

การทำงานของไอซี 8255 จะใช้สัญญาณควบคุมจากไมโครโปรเซสเซอร์ส่งคำสั่งมาจากโปรแกรมการทำงาน หรือกำหนดรูปแบบของพอร์ตให้เป็นอินพุทหรือเอาท์พุทก็ได้

<u>ขาต่าง ๆ ของ 8255</u>

เพื่อให้เข้าใจวิธีการต่อใช้งานระหว่าง Z-80 กับ 8255 จึงเข้าใจความหมายและตำแหน่งของขาต่าง ๆ จำนวน 40 ขา ของไอซีซึ่งประกอบด้วย

D_0-D_7 เป็นขาข้อมูลอินพุทเอาท์พุทจะผ่านเข่าออกทางนี้ D_0-D_7 จึงจะต้องต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์

$\overline{CS}$ ขานี้เป็นขาอินพุทที่จะได้รับสัญญาณจากภายนอกเพื่อเลือกชิพ 8255 โดยขานี้จะแอคตีฟที่ลอจิก "0" จะทำให้ 8255 ต่อเข้ากับ ระบบบัสของไมโครโปรเซสเซอร์เพื่อให้ไมโครโปรเซสเซอร์ เขียนหรืออ่านข้อมูลได้

$\overline{RD}$ เป็นสัญญาณอินพุทที่ส่งมาจากซีพียู เมื่อสัญญาณที่ขานี้เป็น "0" และสัญญาณ เป็น "0" ด้วย ไอซี 8255 จะทำตัวเป็นทางผ่านให้ไมโครโปรเซสเซอร์อ่านข้อมูลจากพอร์ตอินพุทเข้ามา

$\overline{WR}$ เป็นสัญญาณการเขียน จะแอคตีฟเมื่อสัญญาณ $\overline{WR}$ และสัญญาณ $\overline{CS}$ เป็น "0" สัญญาณนี้จะมาจาก ซีพียู เมื่อต้องการเขียนข้อมูลลงพอร์ตที่กำหนด

A_0-A_1 ลอจิกของสัญญานทั้งสองจะถอดรหัสออกเป็น 4 รหัส เพื่อกำหนดรีจีสเตอร์ภายในที่เชื่อมต่อกับพอร์ตอินพุทเอาท์พุทของ 8255

RESET เป็นสัญญาณที่ส่งมาจากภายนอกเข้ามาทำการรีเซต 8255 เพื่อทำการเคลียร์สถานะ ต่าง ๆ ของ 8255

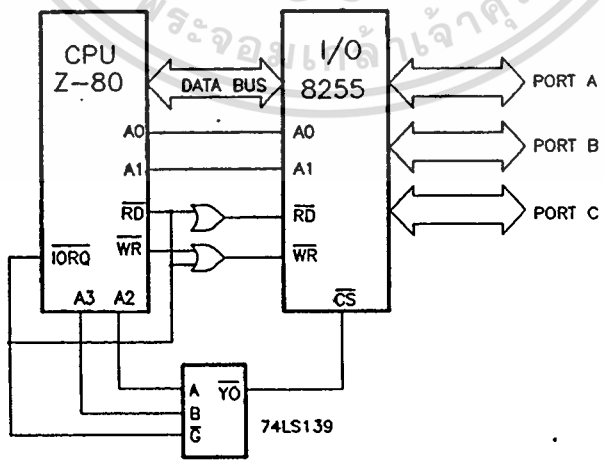
PA_0-PA_7 เป็นสัญญารพอร์ต A ของ 8255 การเลือกพอร์ตนี้ทำได้โดยสัญญาณแอดเดรส A_0-A_1

BA_0-PB_7 เป็นสัญญาณพอร์ต B ของ 8255 การเลือกพอร์ตนี้ทำได้โดยสัญญาณแอดเดรส A_0-A_1

PC_0-PC_7 เป็นสัญญาณพอร์ต C ของ 8255 การเลือกพอร์ตที่ทำได้โดยสัญญาณแอดเดรส A_0-A_1 พอร์ต C แบ่งได้เป็น 2 กลุ่มคือ PC_0-PC_3 และกลุ่ม PC_4-PC_7

<u>และการเชื่อมต่อ Z-80 กับ 8255</u>

หากพิจารณาจากขาต่าง ๆ ของ 8255 จะเห็นได้ว่าส่วนขาควบคุมที่จะเชื่มต่อกับบัสได้ง่าย จากรูปที่ 36 ลองต่อ 8255 เป็นพอร์ต Z-80 และให้ พอร์ต A พอร์ต B และพอร์ต C บนล่าง เป็นพอร์ตหมายเลข 00H,01H,02H และ 03H ตามลำดับ



รูปที่ 36 การเชื่อมต่อ 8255 กับ Z-80

<u>รีจีสเตอร์ภายในของ 8255</u>

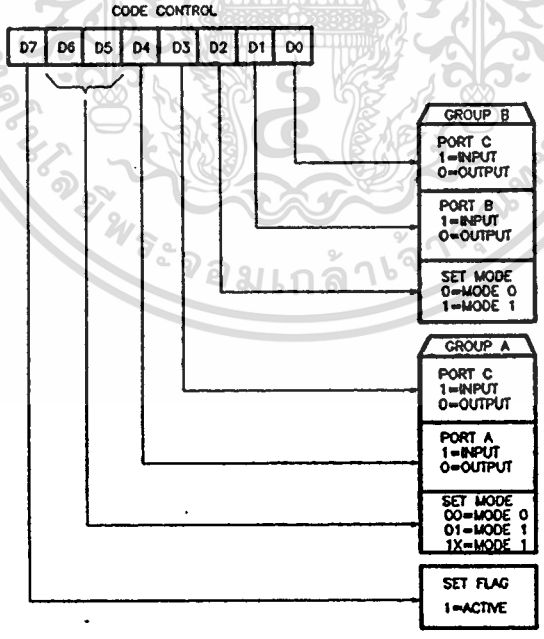
การโปรแกรมให้ 8255 ทำงานตามที่ต้องการจากการที่ 8255 ที่ Z-80 มองเห็น 4 พอร์ต แต่ละพอร์ตจะเสมือนเป็นรีจีสเตอร์ที่สามารถอ่านเขียนได้ รีจีสเตอร์แต่ละตัวนี้จะถูกกำหนดด้วยแอดเรสตามที่ตั้งไว้ เพราะฉนั้นการที่จะให้พอร์ตของ 8255 เป็นอย่างไรขึ้นอยู่ที่ การส่งรหัสควบคุม (control code) ไปควบคุมการทำงานของ 8255 ในที่นี้โดยใช้พอร์ตหมายเลข 03H การควบคุมการทำงานของ 8255 มีทั้งหมด 3 โหมด ในที่นี้จะยกมาให้แค่โหมด 0 เท่านั้น

<u>การทำงานของ 8255 ในโหมด 0</u>

การกำหนดการทำงาน จะส่งข้อมูลที่คำสั่งเข้าโปรแกรมในพอร์ตควบคุมของ 8255 แต่ละบิตที่จะส่งออกไปจะมีความหมายภายในตัวเอง การให้ค่ารหัสบิตต่าง ๆ เข้าไปในรหัสควบคุม แล้วส่งไปยังรีจีสเตอร์ของพอร์ตควบคุม ความหมายของบิตต่าง ๆ มีดังนี้

บิต D_7 เป็นบิตแสดงรหัสควบคุม ถ้าบิตนี้เป็น "1" หมายถึงรหัสควบคุมนี้มีผลต่อการเปลี่ยนแปลงโหมดของ 8255

บิต D_6, D_5 เป็นการเลือกโหมดของ 8255 ซึ่งมี 3 โหมด ดังแสดงในรูปที่ 37



รูปที่ 37 ความหมายของบิตต่าง ๆ ในรหัสควบคุม

บิต D_4 ถ้ามีค่าเป็น "0" หมายถึงการกำหนดพอร์ต A เป็นเอาท์พุทถ้าเป็น "1" จะหมายถึงการกำหนดให้พอร์ต A เป็นอินพุท

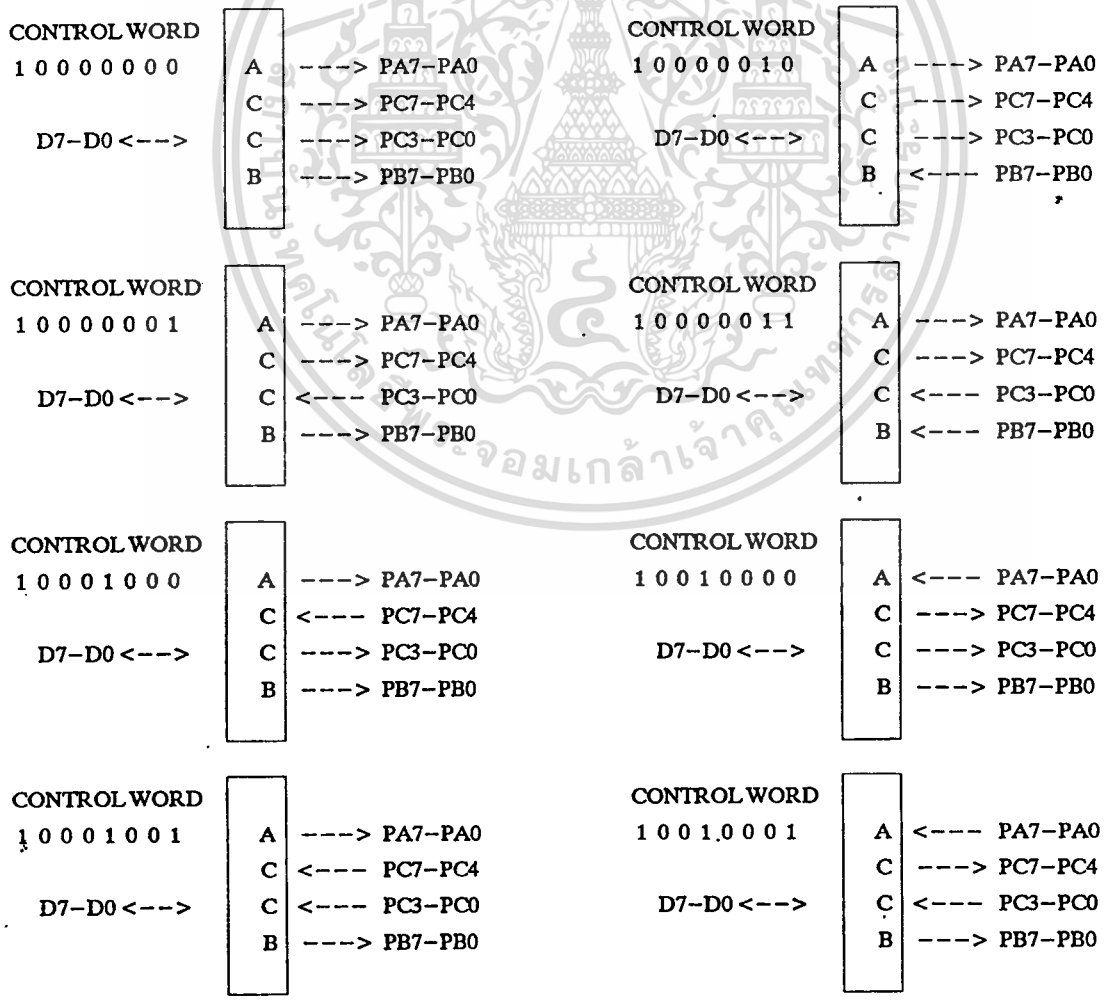
บิต D_3 เป็นบิตที่บอกถึงการเซตของพอร์ต C บนเป็น "0" จะทำให้พอร์ต C เป็นเอาท์พุท

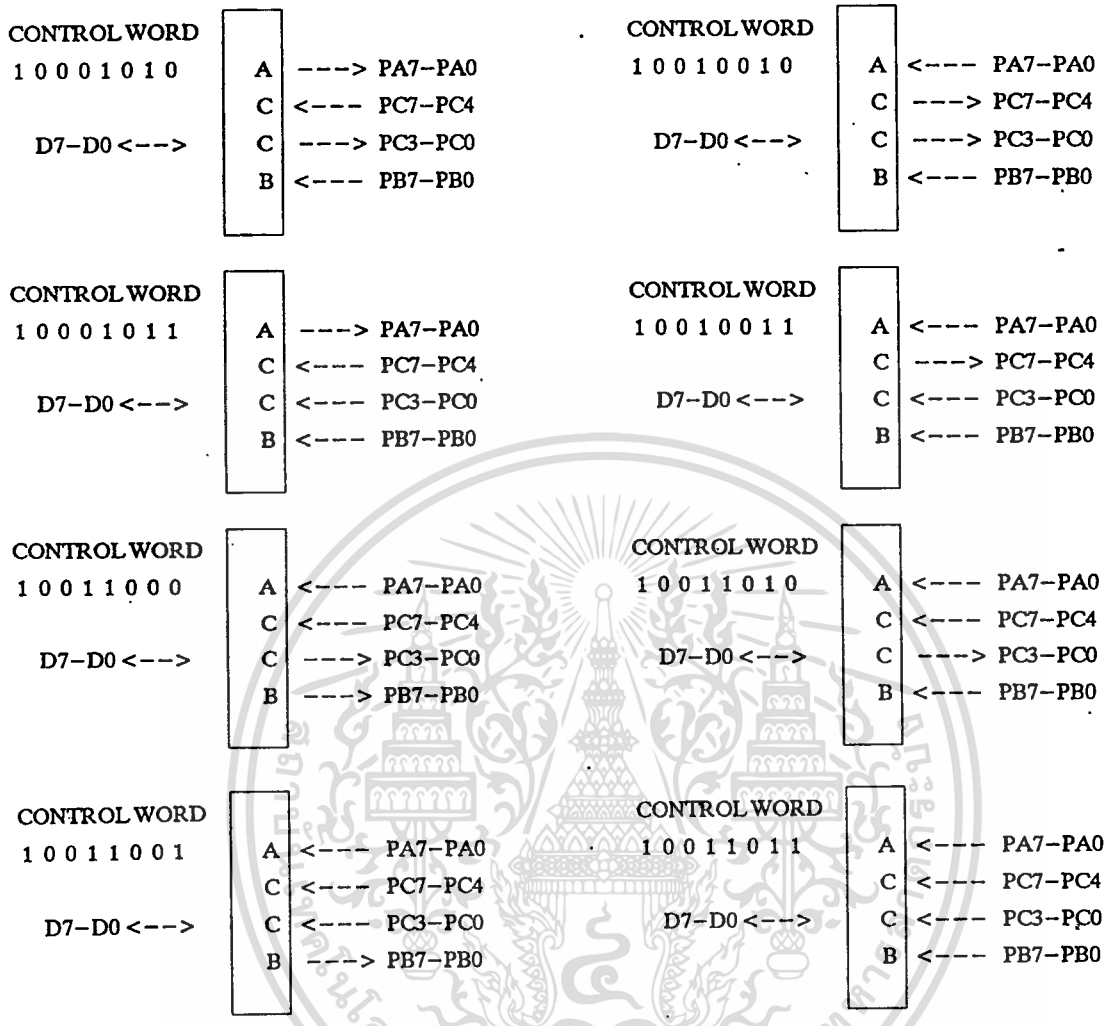
บิต D_2 เป็นบิตที่บอกถึงการเซตโหมดของพอร์ต B ถ้าเป็น "0" หมายถึงการกำหนดให้พอร์ต B ทำงานในโหมด 0

บิต D_1 เป็นบิตที่บอกถึงการทำงานของพอร์ต B ถ้าเป็น "0" หมายถึงกำหนดให้พอร์ต B ทำงานเป็นพอร์ต เอาท์พุท

บิต D_0 เป็นการกำหนดพอร์ตของพอร์ต C ล่าง ถ้าเป็น "0" หมายถึงเป็นเอาท์พุท

ที่กล่าวมาแล้วจะเห็นว่าสามารถที่จะโปรแกรม เลือกรูปแบบพอร์ตได้ทั้งสิ้น 16 แบบ ในโหมด 0



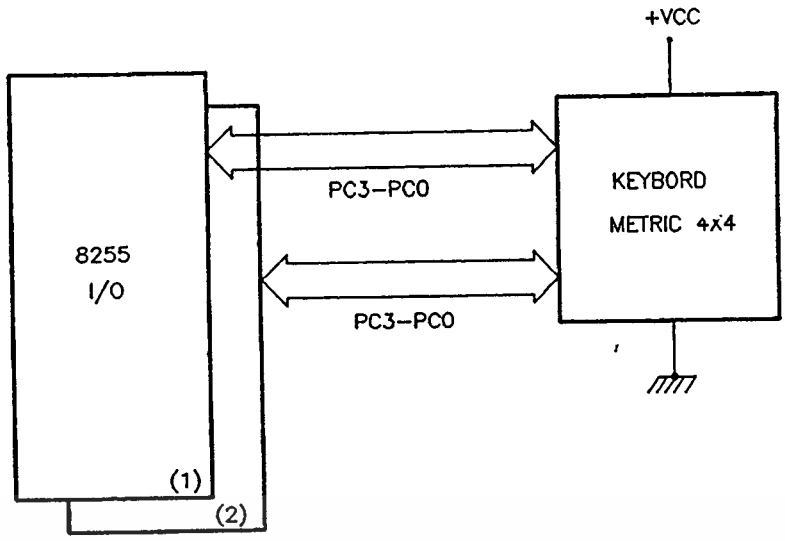


การต่อกับอุปกรณ์ภายนอกกับพอร์ต I/O

การอินเตอร์แฟคอุปกรณ์ภายนอกกับ พอร์ต อินพุท เอาท์พุท สามารถทำได้โดยตรงกับชิป 8255 ซึ่งสามารถกำหนด หรือ โปรแกรมพอร์ตอินพุท เอาท์พุทได้อย่างอิสระ สำหรับในที่นี้แล้วมีอุปกรณ์ 3 ตัวที่จะต้องต่อกับพอร์ต อินพุท/เอาท์พุท คือ Dotmetric display Keybord และ Control output

KEYBORD 4 x 4

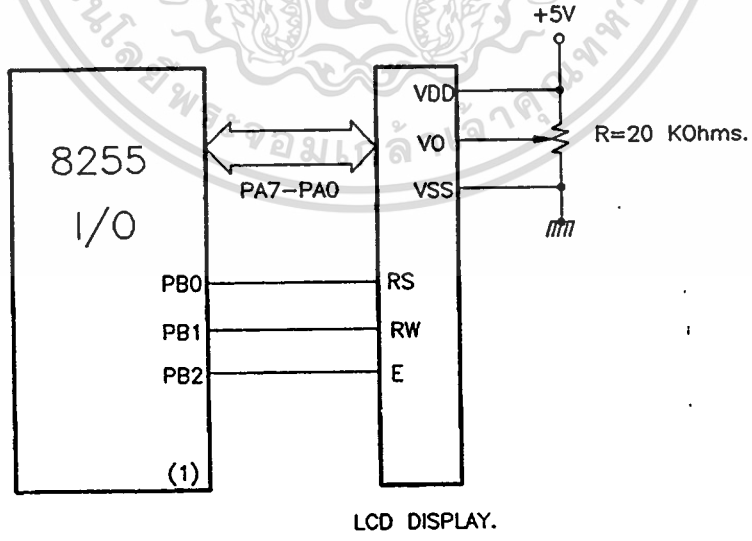
การต่อ Keybord เข้ากับชิป 8255 ทำได้โดยการต่อ เข้ากับ PORT C ของชิป 8255(1) และ 8255(2) คือพอร์ต (02H) , (06H) ตามลำดับดังที่รูป 38



รูปที่ 38 การต่อ Keybord กับชิป 8255

จากรูป จะเห็นได้ว่าการต่อ Keybord กับชิป 8255 ซึ่งไมโครโปรเซสเซอร์ จะมองพอร์ตของชิป 8255 เป็นพอร์ต 02H และ 06H จึงทำให้สามารถ ส่งของมูลออกไปทางพอร์ต 06H และอ่านเข้ามาทางพอร์ต 02H ทำให้สามารถรู้ว่า คีย์ ที่โดนกดอยู่นั้นเป็นคีย์อะไร

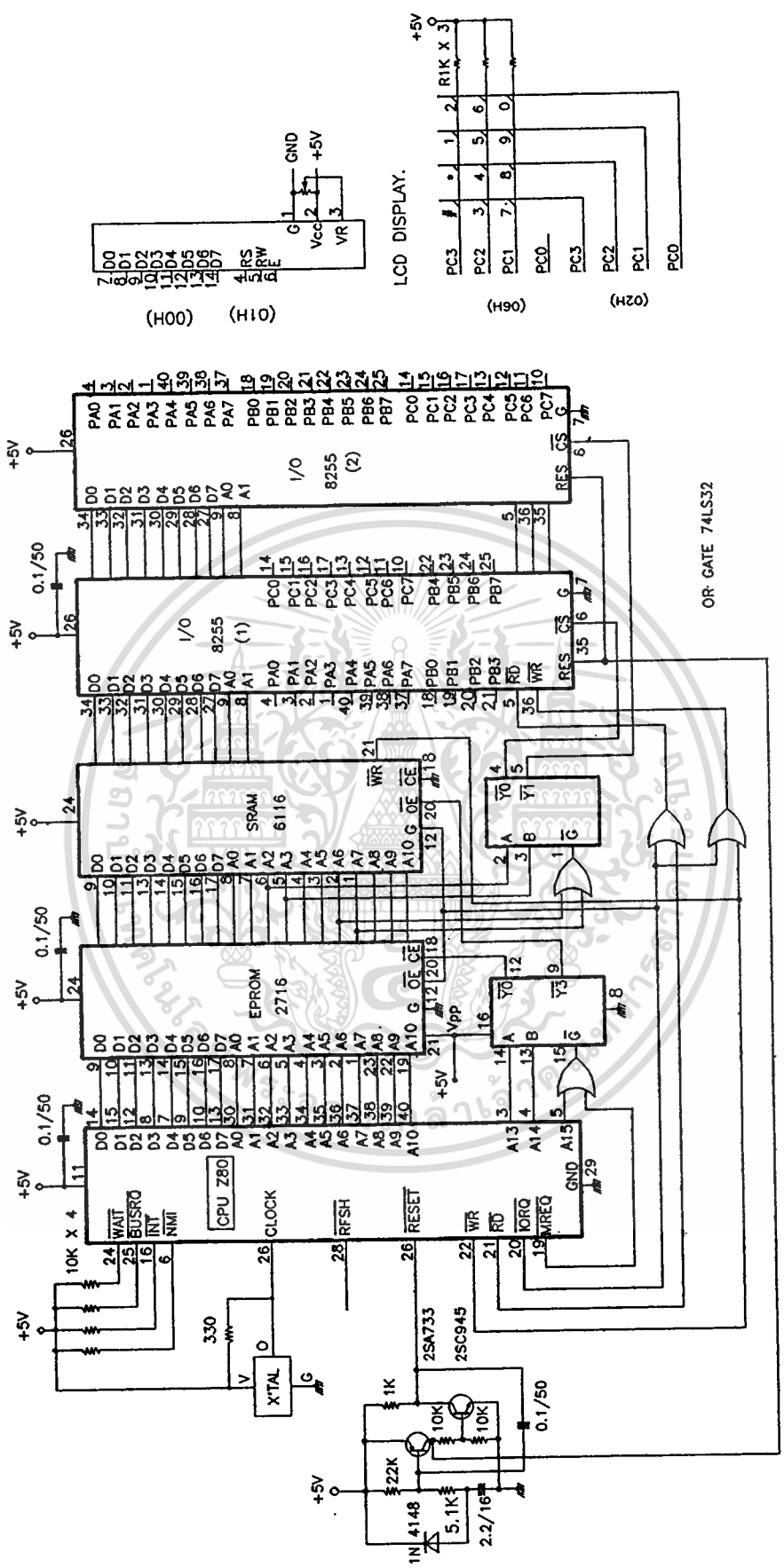
DOTMETRIC DISPLAY



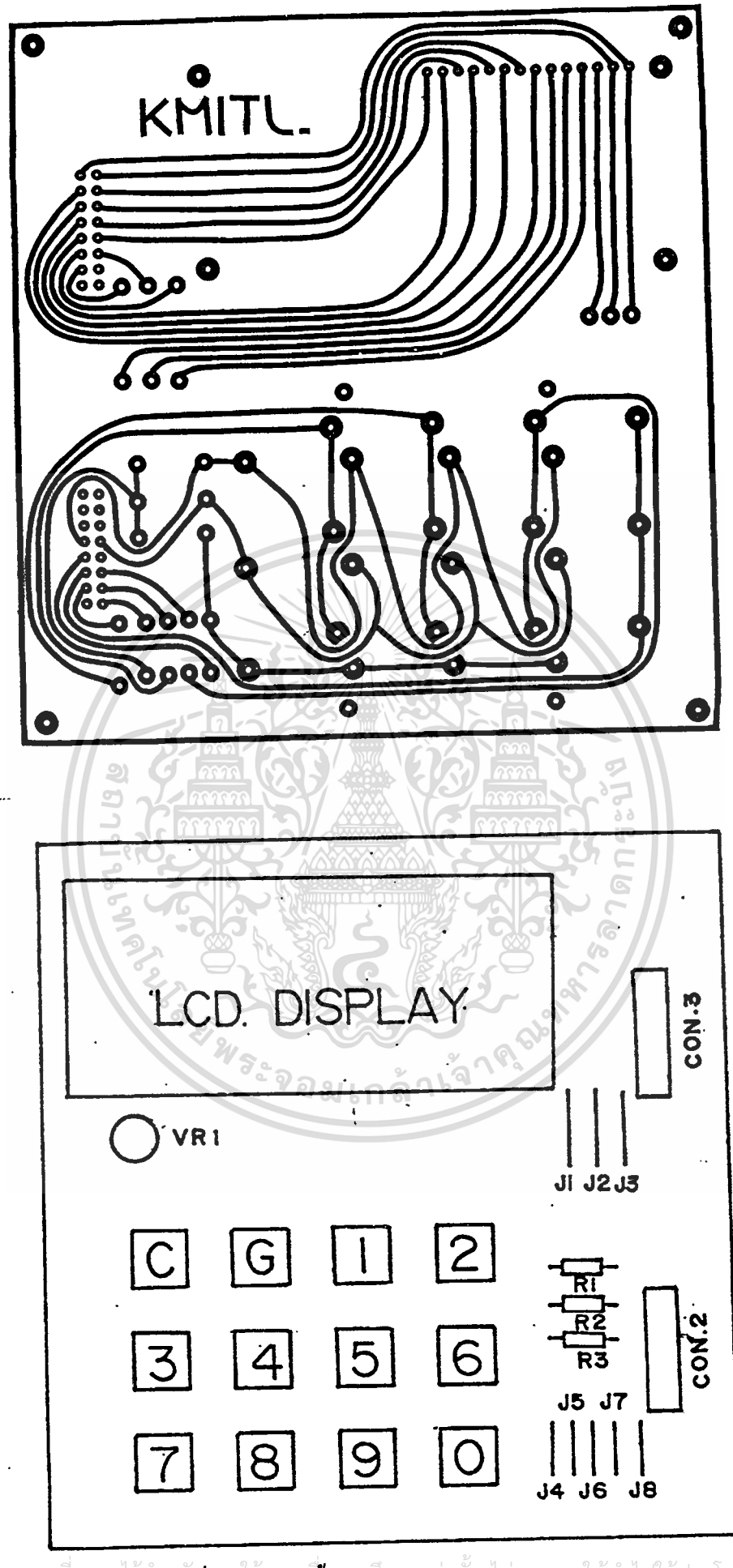
รูปที่ 39 การต่อ LCD display กับชิป 8255

การเชื่อมต่อ Dotmetric LCD display กับชิป 8255 กระทำได้โดยการต่อพอร์ตข้อมูล (D0-D7) ของ LCD Display เข้ากับพอร์ต A (PA_0-PA_7) และต่อ Control ของ LCD Display เข้ากับพอร์ต B ซึ่งใช้แค่ PB_0-PB_2 เพราะ Control ของ LCD Display มีเพียง 3 เส้น คือ RS,RW,E การควบคุม LCD Display มีความสำคัญตรงที่ต้องเข้าใจตัวชิป CONTROLLER ของ LCD Display ในที่นี้ใช้ชิป เบอร์ HD 44780 ซึ่งรายละเอียดคำสั่งควบคุมอยู่ภายในผนวกการต่อ LCD Display กับชิป 8255 ดังรูปที่ 39

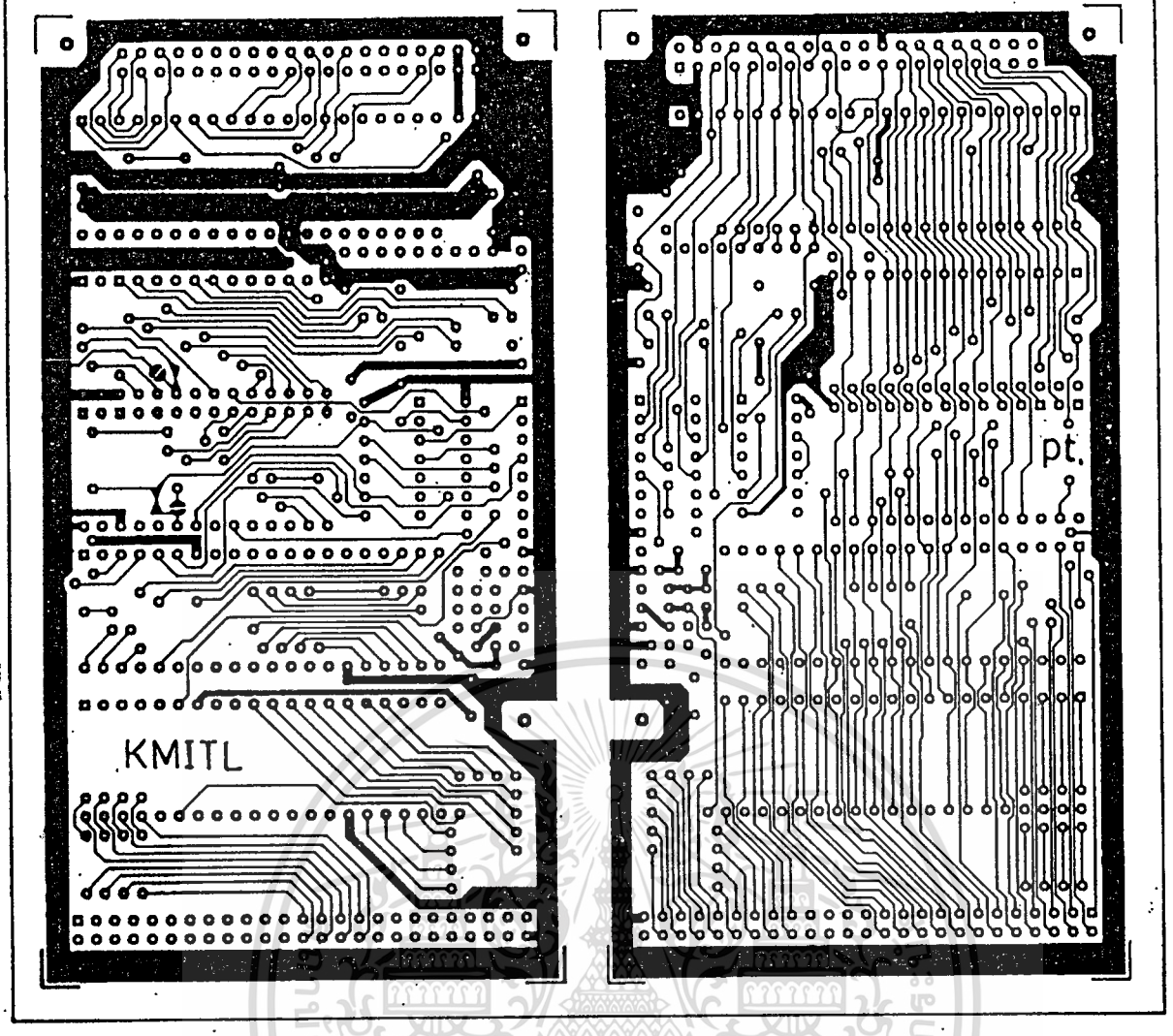
การต่อ LCD Disply โดยการต่อเข้าทางพอร์ต A และ B ของชิป 8255 และใช้คำสั่ง OUT ข้อมูลไปยัง LCD Display โดยที่ข้อมูลได้จาก CHARECTER FONT TABLEในภาคผนวก จำได้อักขระตามที่ต้องการ

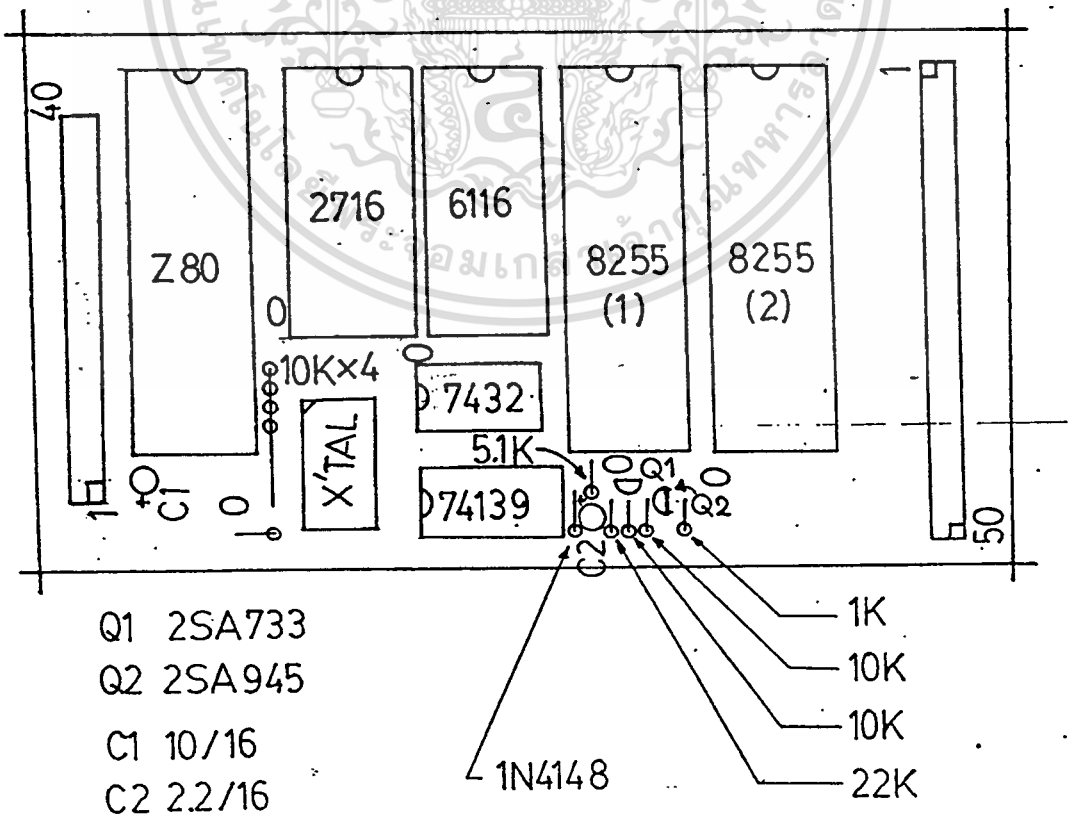


รูปที่ 40 วงจรสมบูรณ์ย์ระทัส

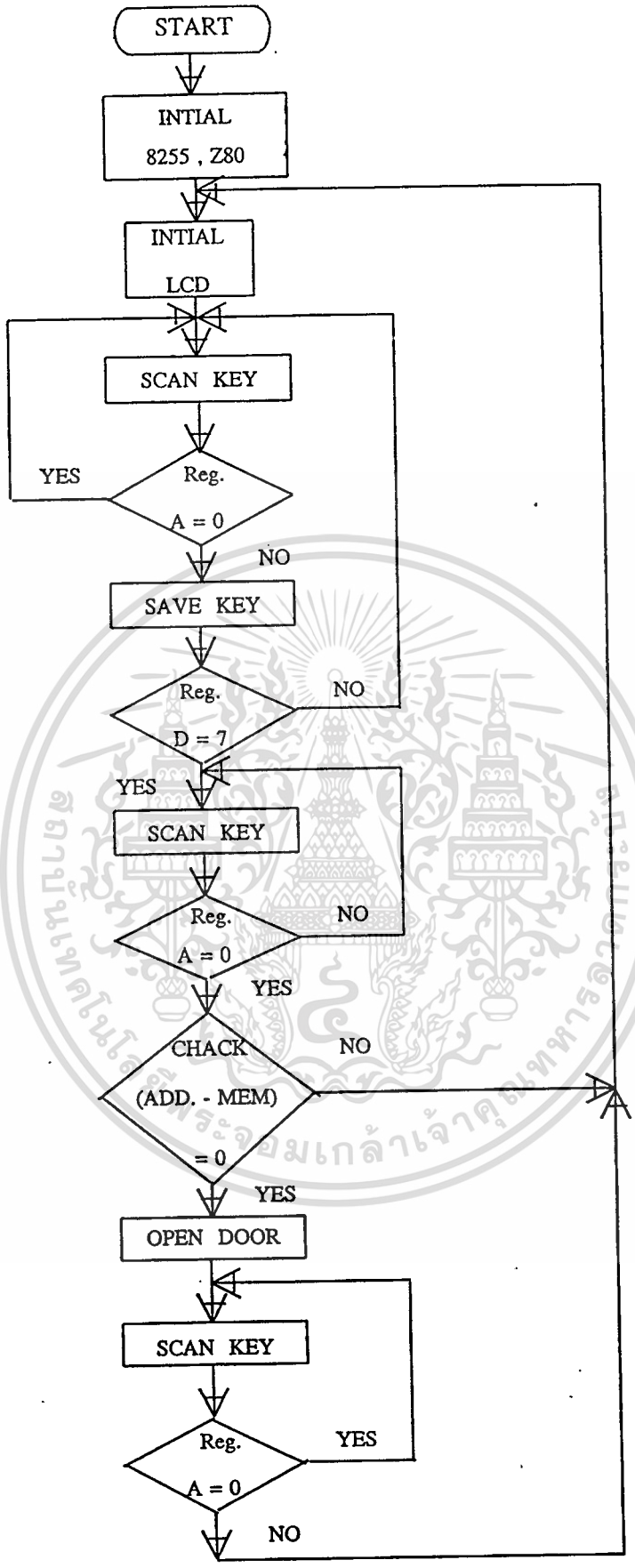


รูปที่ 41 ลายปริ้นวงจรคีย์รหัสและการลงอุปกรณ์





รูปที่ 42 ลายปริ้นวงจรคีย์รหัสและการลงอุปกรณ์



รูปที่ 43 PLOW CHART แสดงการทำงานของเครื่อง

บทที่ 4

วิธีการใช้งาน

1. เริ่มต้น LCD DISPLAY แสดง

PASSWORD:_

2. ผู้ใช้กด คีย์ ระหัส 7 หลัก

PASSWORD:3313121

3. ผู้ใช้กด คีย์ ใด ๆ เป็นคีย์ ENTER เพื่อให้เครื่องประมวลผล
4. ถ้าผู้ใช้กด ระหัส ผิด เครื่องจะให้ผู้ใช้ ไปทำในข้อ 2. ใหม่
5. ถ้าผู้ใช้กด ระหัส ถูก ข้อความที่ปรากฏบน LCD DISPLAY

PASSWORD:<pass>_

พร้อม กับ โซลินอยร์ ทำงาน เปิดประตู

6. เมื่อผู้ใช้ต้องการปิดประตู เพียงกดคีย์ใด ๆ โซลินอยร์ก็จะหยุดทำงานปิดประตู

```
SLRM SuperFast Relocating Macro Assembler    5 Oct 95 01:41 SLRM  1.33 Page   1

KEY     180


   1                    ;*****************************
   2                    ;  SOFTWARE FOR KEY CODE    *
   3                    ;      by TH.Phornchai       *
   4                    ;*****************************
   5          6000      K_PRESS:   EQU   6000H
   6                    ;*********************
   7                    ;  MAIN  PROGRAME    *
   8                    ;*********************
   9          0000      START:     ORG   0000H
  10 0000  AF           XOR   A
  11 0001  31 67FF      LD    SP,67FFH
  12 0004  3E 80        LD    A,80H
  13 0006  D3 03        OUT   (03H),A
  14 0008  3E 89        LD    A,89H
  15 000A  D3 07        OUT   (07H),A
  16 000C  3E 00        LD    A,00H
  17 000E  D3 04        OUT   (04H),A
  18 0010  CD 0000#     MAIN:      CALL  INTIAL
  19 0013  16 00        LD    D,0
  20 0015  1E 00        LD    E,0
  21 0017  CD 0000#     M1:        CALL  SCANK
  22 001A  78           LD    A,B
  23 001B  14           INC   D
  24 001C  CD 0000#     CALL  K_MEM
  25 001F  CD 0000#     CALL  LCD_DISP
```

```
 26 0022  7A                    LD    A,D
 27 0023  FE 07                 CP    07
2928 0025  20 F0                JR    NZ,M1
2929 0027  CD 0000#             CALL  CHACK
2930 002A  18 E4                JR    MAIN
2931                    ;**********************
2932                    ;     SCAN  KEY       *
2933                    ;**********************
2934 002C  CD 0000#     SCANK:  CALL  SCAN
2935 002F  FE FF                CP    0FFH
2936 0031  28 F9                JR    Z,SCANK
2937 0033  CD 0000#             CALL  RELEASE
2938 0036  38 F4                JR    C,SCANK
2939 0038  CD 0000#             CALL  K_CODE
2940 003B  C9                   RET
2941                    ;
2942                       ;/*  scan key  */
2943 003C  06 04        SCAN:   LD    B,4
2944 003E  0E FE                LD    C,0FEH
2945 0040  79           SCAN_1: LD    A,C
7646 0041  D3 02                OUT   (02H),A
7647 0043  DB 06                IN    A,(06H)
7648 0045  E6 0F                AND   0FH
7649 0047  FE 0F                CP    0FH
7650 0049  20 ??                JR    NZ,SCAN_2
7651 004B  CB 01                RLC   C
7652 004D  10 F1                DJNZ  SCAN_1
7653 004F  21 6000              LD    HL,K_PRESS
```

```
7654 0052  CB 86                    RES   0,(HL)
 7655 0054  3E FF                    LD    A,0FFH
43356 0056  C9           SCAN_2:    RET
43357                       ;
43358                       ;/* chack key release  */
43359 0057  21 6000      RELEASE:   LD    HL,K_PRESS
43360 005A  CB 46                   BIT   0,(HL)
43361 005C  37                      SCF
43362 005D  C0                      RET   NZ
43363 005E  CB C6                   SET   0,(HL)
43364 0060  3F                      CCF
43365 0061  C9                      RET
43366 0062  CB 21        K_CODE:    SLA   C
43367 0064  CB 21                   SLA   C
70268 0066  CB 21                   SLA   C
70269 0068  CB 21                   SLA   C
70270 006A  B1                      OR    C
70271 006B  21 0000#                LD    HL,KEYTAB
70272 006E  06 00                   LD    B,0
70273 0070  04           K_CODE1:   INC   B
70274 0071  BE                      CP    (HL)
70275 0072  23                      INC   HL
70276 0073  20 FB                   JR    NZ,K_CODE1
70277 0075  78                      LD    A,B
70278 0076  C9                      RET
70279                       ;
70280                       ; /* intial lcd modul */
70281 0077  CD 0000#     INTIAL:    CALL   DELAY
```

```
70282 007A  CD 0000#          CALL    DELAY
70283 007D  CD 0000#          CALL    DELAY
70284 0080  3E 00             LD      A,00H
70285 0082  D3 00             OUT     (00H),A
70286 0084  D3 01             OUT     (01H),A   ;  RS=0,RW=0
70287 0086  3E 38             LD      A,38H
70288 0088  D3 00             OUT     (00H),A
70289 008A  CD 0000#          CALL    CLOCK1
70290 008D  CD 0000#          CALL    DELAY
70291 0090  3E 0F             LD      A,0FH     ;  D=1,C=1,B=1
70292 0092  D3 00             OUT     (00H),A
70293 0094  CD 0000#          CALL    CLOCK1
70294 0097  CD 0000#          CALL    DELAY
70295 009A  3E 06             LD      A,06H
70296 009C  D3 00             OUT     (00H),A   ;  I/D=1,S=0
70297 009E  CD 0000#          CALL    CLOCK1
70298 00A1  CD 0000#          CALL    DELAY
70299 00A4  3E 01             LD      A,01H
70300 00A6  D3 00             OUT     (00H),A   ;  cursor home
70301 00A8  CD 0000#          CALL    CLOCK1
70302 00AB  CD 0000#          CALL    DELAY
70303 00AE  3E 01             LD      A,01H
70304 00B0  D3 01             OUT     (01H),A   ;  RS=1,RW=0
70305 00B2  3E 50             LD      A,50H
70306 00B4  D3 00             OUT     (00H),A   ;  "P"
70307 00B6  CD 0000#          CALL    CLOCK2
70308 00B9  3E 41             LD      A,41H
70309 00BB  D3 00             OUT     (00H),A   ;  "A"
```

```
70310 00BD  CD 0000#          CALL   CLOCK2
70311 00C0  3E 53             LD     A,53H
70312 00C2  D3 00             OUT    (00H),A  ;  "S"
70313 00C4  CD 0000#          CALL   CLOCK2
70314 00C7  3E 53             LD     A,53H
70315 00C9  D3 00             OUT    (00H),A  ;  "S"
70316 00CB  CD 0000#          CALL   CLOCK2
70317 00CE  3E 57             LD     A,57H
70318 00D0  D3 00             OUT    (00H),A  ;  "W"
70319 00D2  CD 0000#          CALL   CLOCK2
70320 00D5  3E 4F             LD     A,4FH
70321 00D7  D3 00             OUT    (00H),A  ;  "O"
70322 00D9  CD 0000#          CALL   CLOCK2
70323 00DC  3E 52             LD     A,52H
70324 00DE  D3 00             OUT    (00H),A  ;  "R"
70325 00E0  CD 0000#          CALL   CLOCK2
70326 00E3  3E 44             LD     A,44H
70327 00E5  D3 00             OUT    (00H),A  ;  "D"
70328 00E7  CD 0000#          CALL   CLOCK2
70329 00EA  3E C0             LD     A,0C0H
70330 00EC  D3 00             OUT    (00H),A
70331 00EE  CD 0000#          CALL   CLOCK1
70332 00F1  3E 3A             LD     A,3AH
70333 00F3  D3 00             OUT    (00H),A  ;  ":"
70334 00F5  CD 0000#          CALL   CLOCK2
70335 00F8  C9                RET
70336                   ;
70337                   ;/* pulse for enable */
```

```
70338 00F9  3E 00       CLOCK1:   LD      A,00H
70339 00FB  D3 01                 OUT     (01H),A
70340 00FD  CD 0000#              CALL    DELAY
70341 0100  3E 04                 LD      A,04H     ;  clock
70342 0102  D3 01                 OUT     (01H),A
70343 0104  CD 0000#              CALL    DELAY
70344 0107  3E 00                 LD      A,00H
70345 0109  D3 01                 OUT     (01H),A
70346 010B  C9                    RET
70347 010C  3E 01       CLOCK2:   LD      A,01H
70348 010E  D3 01                 OUT     (01H),A
70349 0110  CD 0000#              CALL    DELAY
70350 0113  3E 05                 LD      A,05H     ;  clock
70351 0115  D3 01                 OUT     (01H),A
70352 0117  CD 0000#              CALL    DELAY
70353 011A  3E 01                 LD      A,01H
70354 011C  D3 01                 OUT     (01H),A
70355 011E  C9                    RET
70356                     ;
70357                     ;/* delay time */
70358 011F  26 6F       DELAY:    LD      H,06FH    ;  delay
70359 0121  2E FF       DE2:      LD      L,0FFH
70360 0123  2D          DE1:      DEC     L
70361 0124  20 FD                 JR      NZ,DE1
70362 0126  25                    DEC     H
70363 0127  20 F8                 JR      NZ,DE2
70364 0129  C9                    RET
70365                     ;
```

```
70366                          ;/* save key */
70367 012A  1C          K_MEM:   INC    E
70368 012B  7B                   LD     A,E
70369 012C  FE 01                CP     1
70370 012E  CC 0000#             CALL   Z,K1
70371 0131  7B                   LD     A,E
70372 0132  FE 02                CP     2
70373 0134  CC 0000#             CALL   Z,K2
70374 0137  7B                   LD     A,E
70375 0138  FE 03                CP     3
70376 013A  CC 0000#             CALL   Z,K3
70377 013D  7B                   LD     A,E
70378 013E  FE 04                CP     4
70379 0140  CC 0000#             CALL   Z,K4
70380 0143  7B                   LD     A,E
70381 0144  FE 05                CP     5
78682 0146  CC 0000#             CALL   Z,K5
78683 0149  7B                   LD     A,E
78684 014A  FE 06                CP     6
78685 014C  CC 0000#             CALL   Z,K6
78686 014F  7B                   LD     A,E
78687 0150  FE 07                CP     7
3=5>8 0152  CC 0000#             CALL   Z,K7
98549 0155  C9                   RET
98550 0156  78          K1:      LD     A,B
98551 0157  32 6600              LD     (6600H),A
98552 015A  C9                   RET
98553 015B  78          K2:      LD     A,B
```

```
4054 015C  32 6601            LD     (6601H),A
4055 015F  C9                 RET
4056 0160  78          K3:    LD     A,B
4057 0161  32 6602            LD     (6602H),A
4058 0164  C9                 RET
4059 0165  78          K4:    LD     A,B
4060 0166  32 6603            LD     (6603H),A
4061 0169  C9                 RET
4062 016A  78          K5:    LD     A,B
4063 016B  32 6604            LD     (6604H),A
4064 016E  C9                 RET
4065 016F  78          K6:    LD     A,B
4066 0170  32 6605            LD     (6605H),A
4067 0173  C9                 RET
4068 0174  78          K7:    LD     A,B
4069 0175  32 6606            LD     (6606H),A
4070 0178  C9                 RET
4071                   ;
4072                   ;/* lcd display */
4073 0179  78          LCD_DISP: LD   A,B
4074 017A  FE 01              CP     1
4075 017C  CC 0000#           CALL   Z,D1
4076 017F  78                 LD     A,B
4077 0180  FE 02              CP     2
4078 0182  CC 0000#           CALL   Z,D2
4079 0185  78                 LD     A,B
4080 0186  FE 03              CP     3
4081 0188  CC 0000#           CALL   Z,D3
```

```
4082 018B  78          LD    A,B
4083 018C  FE 04       CP    4
4084 018E  CC 0000#    CALL  Z,D4
4085 0191  78          LD    A,B
4086 0192  FE 05       CP    5
4087 0194  CC 0000#    CALL  Z,D5
4088 0197  78          LD    A,B
4089 0198  FE 06       CP    6
4090 019A  CC 0000#    CALL  Z,D6
4091 019D  78          LD    A,B
4932 019E  FE 07       CP    7
24753 01A0 CC 0000#    CALL  Z,D7
24754 01A3 78          LD    A,B
24755 01A4 FE 08       CP    8
24756 01A6 CC 0000#    CALL  Z,D8
24757 01A9 78          LD    A,B
24758 01AA FE 09       CP    9
24759 01AC CC 0000#    CALL  Z,D9
24760 01AF 78          LD    A,B
24761 01B0 FE 0A       CP    10
24762 01B2 CC 0000#    CALL  Z,D10
24763 01B5 78          LD    A,B
24764 01B6 FE 0B       CP    11
24765 01B8 CC 0000#    CALL  Z,D11
24766 01BB 78          LD    A,B
24767 01BC FE 0C       CP    12
24768 01BE CC 0000#    CALL  Z,D12
24769 01C1 C9          RET
```

```
24770 01C2  3E 23       D1:     LD      A,23H       ; #
24771 01C4  D3 00               OUT     (00H),A
24772 01C6  CD 010C             CALL    CLOCK2
24773 01C9  C9                  RET
24774 01CA  3E 2A       D2:     LD      A,2AH       ; *
24775 01CC  D3 00               OUT     (00H),A
24776 01CE  CD 010C             CALL    CLOCK2
24777 01D1  C9                  RET
24778 01D2  3E 31       D3:     LD      A,31H       ; 1
24779 01D4  D3 00               OUT     (00H),A
24780 01D6  CD 010C             CALL    CLOCK2
289>1 01D9  C9                  RET
53242 01DA  3E 32       D4:     LD      A,32H       ; 2
53243 01DC  D3 00               OUT     (00H),A
53244 01DE  CD 010C             CALL    CLOCK2
53245 01E1  C9                  RET
53246 01E2  3E 33       D5:     LD      A,33H       ; 3
53247 01E4  D3 00               OUT     (00H),A
53248 01E6  CD 010C             CALL    CLOCK2
53249 01E9  C9                  RET
53250 01EA  3E 34       D6:     LD      A,34H       ; 4
53251 01EC  D3 00               OUT     (00H),A
53252 01EE  CD 010C             CALL    CLOCK2
53253 01F1  C9                  RET
53254 01F2  3E 35       D7:     LD      A,35H       ; 5
53255 01F4  D3 00               OUT     (00H),A
53256 01F6  CD 010C             CALL    CLOCK2
53257 01F9  C9                  RET
```

```
53258 01FA  3E 36      D8:      LD     A,36H      ; 6
53259 01FC  D3 00               OUT    (00H),A
53260 01FE  CD 010C             CALL   CLOCK2
53261 0201  C9                  RET
53262 0202  3E 37      D9:      LD     A,37H      ; 7
53263 0204  D3 00               OUT    (00H),A
53264 0206  CD 010C             CALL   CLOCK2
53265 0209  C9                  RET
53266 020A  3E 38      D10:     LD     A,38H      ; 8
53267 020C  D3 00               OUT    (00H),A
53268 020E  CD 010C             CALL   CLOCK2
53269 0211  C9                  RET
53270 0212  3E 39      D11:     LD     A,39H      ; 9
53271 0214  D3 00               OUT    (00H),A
53272 0216  CD 010C             CALL   CLOCK2
53273 0219  C9                  RET
53274 021A  3E 30      D12:     LD     A,30H      ; 0
53275 021C  D3 00               OUT    (00H),A
53276 021E  CD 010C             CALL   CLOCK2
53277 0221  C9                  RET
53278 0222  CD 003C    CHACK:   CALL   SCAN
53279 0225  FE FF               CP     0FFH
53280 0227  28 F9               JR     Z,CHACK
53281 0229  CD 0057             CALL   RELEASE
53282 022C  38 F4               JR     C,CHACK
53283 022E  3A 6600             LD     A,(6600H)
53284 0231  FE 05               CP     5
53285 0233  20 ??               JR     NZ,JOP
```

```
53286 0235  3A 6601          LD      A,(6601H)
53287 0238  FE 05            CP      5
53288 023A  20 ??            JR      NZ,JOP
53289 023C  3A 6602          LD      A,(6602H)
53290 023F  FE 03            CP      3
53291 0241  20 ??            JR      NZ,JOP
53292 0243  3A 6603          LD      A,(6603H)
53293 0246  FE 05            CP      5
53294 0248  20 ??            JR      NZ,JOP
53295 024A  3A 6604          LD      A,(6604H)
53296 024D  FE 03            CP      3
53297 024F  20 ??            JR      NZ,JOP
53298 0251  3A 6605          LD      A,(6605H)
89279 0254  FE 04            CP      4
25360 0256  20 ??            JR      NZ,JOP
25361 0258  3A 6606          LD      A,(6606H)
25362 025B  FE 03            CP      3
25363 025D  20 ??            JR      NZ,JOP
25364 025F  CD 0077          CALL    INTIAL
25365 0262  3E 3C            LD      A,3CH       ; <
25366 0264  D3 00            OUT     (00H),A
25367 0266  CD 010C          CALL    CLOCK2
25368 0269  3E 70            LD      A,70H       ; p
25369 026B  D3 00            OUT     (00H),A
25370 026D  CD 010C          CALL    CLOCK2
25371 0270  3E 61            LD      A,61H       ; a
25372 0272  D3 00            OUT     (00H),A
25373 0274  CD 010C          CALL    CLOCK2
```

```
25374 0277  3E 73                    LD      A,73H       ; s
25375 0279  D3 00                    OUT     (00H),A
25376 027B  CD 010C                  CALL    CLOCK2
25377 027E  3E 73                    LD      A,73H       ; s
25378 0280  D3 00                    OUT     (00H),A
25379 0282  CD 010C                  CALL    CLOCK2
25380 0285  3E 3E                    LD      A,3EH       ; >
25381 0287  D3 00                    OUT     (00H),A
25382 0289  CD 010C                  CALL    CLOCK2
25383 028C  3E FF                    LD      A,0FFH      ;key code correct
25384 028E  D3 04                    OUT     (04H),A
25385 0290  CD 003C      LOOP:       CALL    SCAN
25386 0293  FE FF                    CP      0FFH
25387 0295  28 F9                    JR      Z,LOOP
25388 0297  3E 00                    LD      A,00H
25389 0299  D3 04                    OUT     (04H),A
25390 029B  CD 0057                  CALL    RELEASE
25391 029E  38 F0                    JR      C,LOOP
25392 02A0  C9           JOP:        RET
25393                        ;
25394                            ;/*  table key bord  */
25395 02A1  77 B7 D7 E7 KEYTAB:      DB   077H,0B7H,0D7H,0E7H  ;key #,*,1,2
25396 02A5  7B BB DB EB              DB   07BH,0BBH,0DBH,0EBH  ;    3,4,5,6
25397 02A9  7D BD DD ED              DB   07DH,0BDH,0DDH,0EDH  ;    7,8,9,0
 0 Error(s) Detected.
 685 Absolute Bytes. 42 Symbols Detected.
```

ภาคผนวก

ตารางคำสั่ง HD44780

Instruction	Code										Description	Execution time (when fosc is 250 kHz) Note 1	Execution time (when fosc is 160 kHz) Note 2
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	82 μs ~ 1.64 ms	120 μs ~ 4.9 ms
Return home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	40 μs ~ 1.6 ms	120 μs ~ 4.8 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μs	120 μs
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μs	120 μs
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents	40 μs	120 μs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).	40 μs	120 μs
Set CG RAM address.	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs	120 μs
Set DD RAM address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs	120 μs
Read busy flag & address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	1μs	1 μs
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs	120 μs
Read data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs	120 μs

I/D = 1: Increment (+1) I/D = 0: Decrement (−1)
S = 1: Accompanies display shift.
S/C = 1: Display shift S/C = 0: Cursor move
R/L = 1: Shift to the right.
R/L = 0: Shift to the left.
DL = 1: 8 bits DL = 0: 4 bits
N = 1: 2 lines N = 0: 1 line
F = 1: 5 x 10 dots F = 0: 5 x 7 dots
BF = 1: Internally operating
BF = 0: Can accept instruction

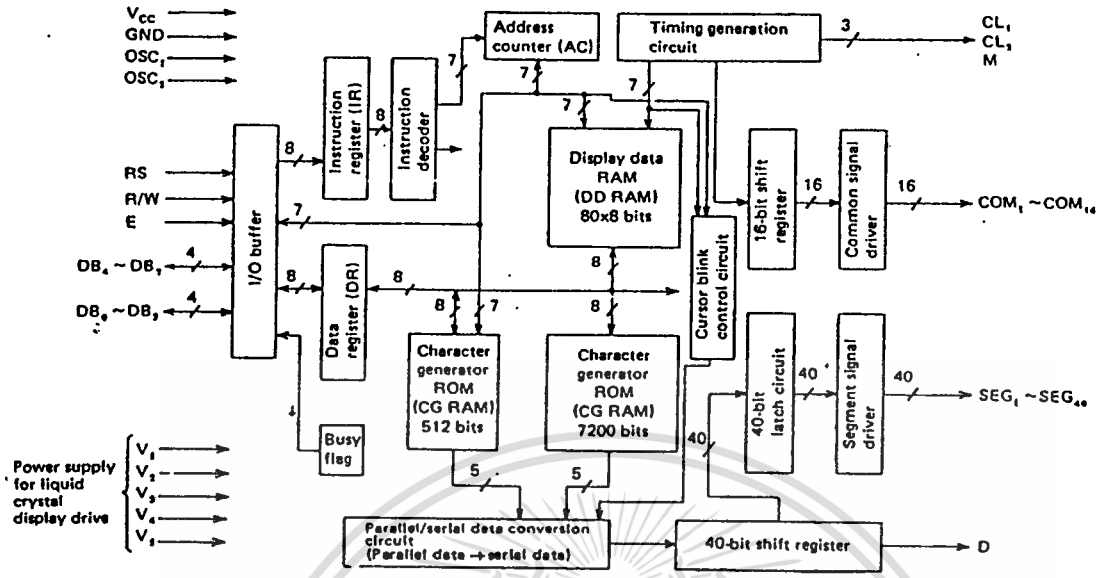
DD RAM: Display data RAM
CG RAM: Character generator RAM
A_{CG}: CG RAM address
A_{DD}: DD RAM address Corresponds to cursor address.
AC: Address counter used for both of DD and CG RAM address.

Execution time changes when frequency changes.
(Example)
When fosc is 270 kHz:

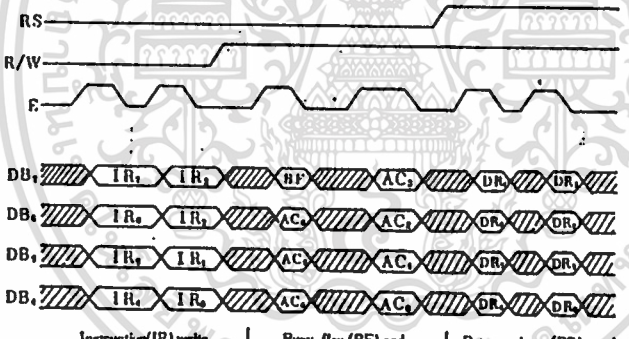
$$40\ \mu s \times \frac{250}{270} = 37\ \mu s$$

*No effect
Notes 1. Applied to models driven by 1/8 duty or 1/11 duty.
2. Applied to models driven by 1/16 duty.

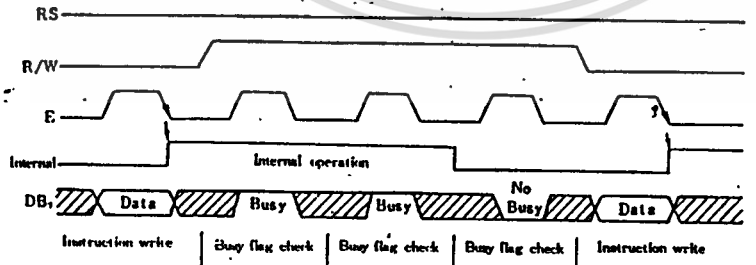
Block diagram of HD44780 interior



เท่านั้นโดยข้อมูลครั้งแรกที่ส่งนั้น HD44780 จะถือเป็นข้อมูล 4 BIT บน และข้อมูลที่ส่งต่อมานั้นเ
เป็นข้อมูล 4 BIT ล่าง



4-bit data transfer example



Example of busy flag check timing sequence

ขาต่างๆในการต่อใช้งาน HD44780

1. RS (REGISTOR SELECTION) จะเป็นขาเลือก REGISTOR ภายในซึ่งมีอยู่ 2 ตัวคือ INSTRUCTION REGISTOR (IR) และ DATA REGISTOR (DR) โดยถ้าเป็น 1 จะเป็นการเลือก DATA และถ้าเป็น 0 จะเป็นการเลือก INSTRUCTION

2. R/W (READ/WRITE) เป็นตัวเลือกว่าจะเขียนหรือจะอ่านข้อมูลจากตัว IC โดยอ่านข้อมูล = 1, เขียนข้อมูล = 0

3. E (ENABLE SIGNAL) เป็นขากำหนดสภาพการรับเขียนอ่านข้อมูล

The relation between the operation and the combination of RS, R/W

RS	RW	E	OPERATION
0	0		Write instruction code
0	1		Read busy flag and address counter
1	0		Write data
1	1		Read data

When performing data and instruction code by 4 bit, transfer RS, R/W every time.

4. DB0-DB7 เป็นขารับส่งข้อมูลจากตัว IC

5. VDD ไฟเลี้ยงตัววงจร

6. VSS เป็นขา GND

7. VO เป็นขารับ VOLTAGE ในการขับ LCD ให้สว่างหรือมืด

รายละเอียดของคำสั่ง HD44780

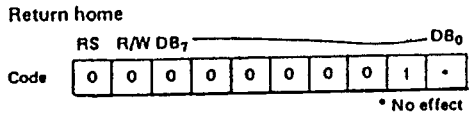
1. CLEAR DISPLAY

Clear display

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	0	0	0	1

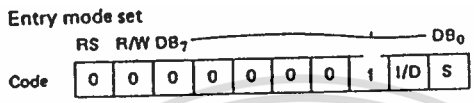
คำสั่งนี้จะเป็นการเขียนช่องว่างหรือ SPACE (ASCII 20H) เข้าไปใน DD RAM ทั้งหมดและทำการ SET DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพ SET I/D = 1, S ไม่มีการเปลี่ยน

2. RETURN HOME



คำสั่งนี้จะทำการ SET DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพข้อมูลในจอภาพไม่เปลี่ยน

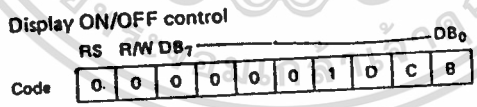
3. ENTRY MODE SET



BIT I/D : โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียนหรืออ่านข้อมูลแล้วจะทำให้ DD RAM ADDRESS เพิ่มขึ้นหนึ่งหรือลดลงหนึ่งโดย 1 = เพิ่ม 0 = ลดลงหนึ่ง

BIT S : เป็นตัวกำหนดแสดงผลโดยถ้า S = 1 จะเป็นการใส่ข้อมูลแล้วตัว CURSOR อยู่กับที่ข้อมูลจะถูกเคลื่อนไปทางซ้าย ถ้า S = 0 ข้อมูลจะอยู่กับที่ตัว CURSOR จะถูกเคลื่อนไปทางขวามือ

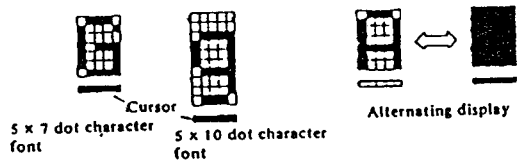
4. DISPLAY ON/OFF CONTROL



BIT D : เป็น BIT ให้เปิดปิดหน้าจอภาพโดยถ้า D = 1 จะ ON และ D = 0 จะ OFF

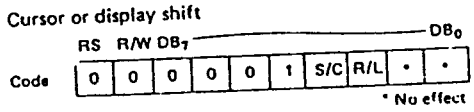
BIT C : จะให้แสดง CURSOR ให้ BIT C = 1 และถ้าไม่ต้องการแสดง CURSOR BIT C = 0 โดยตัว CURSOR จะอยู่ที่ LINE ที่ 8 ในแบบ 5X7 DOT และจะอยู่ LINE ที่ 11 ในแบบ 5X10 DOT

BIT B : เป็น BIT SET การกระพริบของ CURSOR โดย B = 1 มีการกระพริบ B = 0 ไม่มีการกระพริบ โดยมีระยะเวลาการกระพริบประมาณ 379.2 ms



(a) Cursor Display Example (b) Blink Display Example

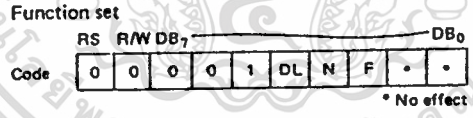
5. CURSOR OR DISPLAY SHIFT



เป็นคำสั่งกำหนดให้ตำแหน่ง CURSOR หรือข้อมูลไปเกิดทางซ้ายหรือขวา โดยไม่ต้องใช้คำสั่งเขียนหรืออ่าน โดย

S/C	R/L	
0	0	ทำการย้าย CURSOR ไปจากตำแหน่งเดิมไปซ้ายมือ 1 ตำแหน่ง
0	1	ทำการย้าย CURSOR ไปจากตำแหน่งเดิมไปขวามือ 1 ตำแหน่ง
1	0	เป็นการดันตัวอักษรที่เกิดไปทางซ้าย
1	1	เป็นการดันตัวอักษรที่เกิดไปทางขวามือ

6. FUNCTION SET



BIT DL : เป็นการ SET การติดต่อว่าจะให้เป็นแบบ 8 BIT หรือ 4 BIT โดยถ้าต้องการติดต่อ 4 BIT DL = 0 และ 8 BIT DL = 1

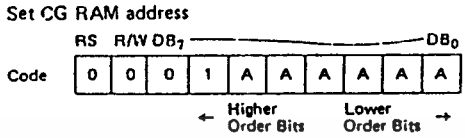
N : เป็นการ SET บรรทัดการแสดงผล N = 0 แสดง 1 บรรทัด N = 1 แสดง 2 บรรทัด ในกรณีมากกว่า 2 บรรทัด ก็ให้ SET N = 1

F : เป็นการ SET ขนาด DOT การแสดงผล 5X7 หรือ 5X10 โดย F = 0 เป็นแบบ 5X7 และ F = 1 เป็นแบบ 5X10

N F	No. of display lines	Character font	Duty factor	Remarks
0 0	1	5 x 7 dots	1/8	
0 1	1	5 x 10 dots	1/11	
1 *	2	5 x 7 dots	1/16	Cannot display 2 lines with 5 x 10 dot character font.

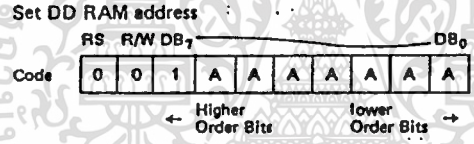
* No effect

7. SET CG RAM ADDRESS



ใน HD44780 นั้นจะมีหน่วยความจำอยู่ 2 ชุด คือ DISPLAY DATA RAM (DD RAM) จำนวน 80X8 BIT และ CHARACTER GENERATOR ROM · CG RAM จำนวน 512 BIT และ 7200 BIT คำสั่งนี้จะเป็นการ SET ADDRESS ใน CG RAM โดยต้องทำการ SET ADDRESS ก่อนเขียนหรืออ่านข้อมูลจาก CG RAM ด้วย

8. SET DD RAM ADDRESS



เป็นคำสั่ง SET ค่า ADDRESS ใน DD RAM ในการเขียนหรืออ่านค่าจาก DD RAM (DD RAM คือ ส่วนที่จะแสดงผลหน้าจอ LCD) โดยจำนวน ADDRESS ที่จะเกิดขึ้นบนจอ LCD จะอยู่กับ SET ค่า N ด้วย

ถ้า N = 0 (1 บรรทัด) ADDRESS จะอยู่ 00H-4FH

ถ้า N = 1 (2 บรรทัด) ADDRESS จะอยู่ 00H-27H สำหรับบรรทัดที่ 1 และ 40H-67H สำหรับบรรทัดที่ 2

ตัวอย่างการจัด ADDRESS ของ DD RAM หน้าจอ LCD แบบ 16 ตัวอักษร 4 บรรทัด และ 20 ตัวอักษร 2 บรรทัด HDM-16416H, HDM-20216H

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	← display position
1-line	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	← DD RAM address
2-line	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	
3-line	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
4-line	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	

HDM-16416H

	1ที่	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	←display position
1-line	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	← DD RAM address
2-line	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	
3-line	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	
4-line	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	.61	62	63	64	65	66	67	

(Note)Shift display is as same as 2-line type.

HDM-20216H

9. READ BUSY FLAG AND ADDRESS

Read busy flag & address

	RS	R/W	DB_7						DB_0
Code	0	1	BF	A	A	A	A	A	A	A

← Higher Order Bits Lower Order Bits →

เป็นคำสั่งอ่านค่า BUSY FLAG ซึ่งจะเป็นตัวบอกว่าตัว HD44780 นี้อยู่ในขบวนการทำงานภายในอยู่หรืออยู่ในสภาพพร้อมจะรับข้อมูล โดย

BF = 1 อยู่ในขบวนการทำงานภายในไม่พร้อมจะรับข้อมูลหรือคำสั่ง

BF = 0 พร้อมจะรับข้อมูลหรือคำสั่งได้

และนอกจากนี้ยังเป็นคำสั่งอ่านค่าข้อมูล ADDRESS ของ CG RAM หรือ DD RAM ด้วย

10. WRITE DATA TO CG หรือ DD RAM

Write data to CG or DD RAM

	RS	R/W	DB_7						DB_0
Code	1	0	D	D	D	D	D	D	D	D

← Higher Order Bits Lower Order Bits →

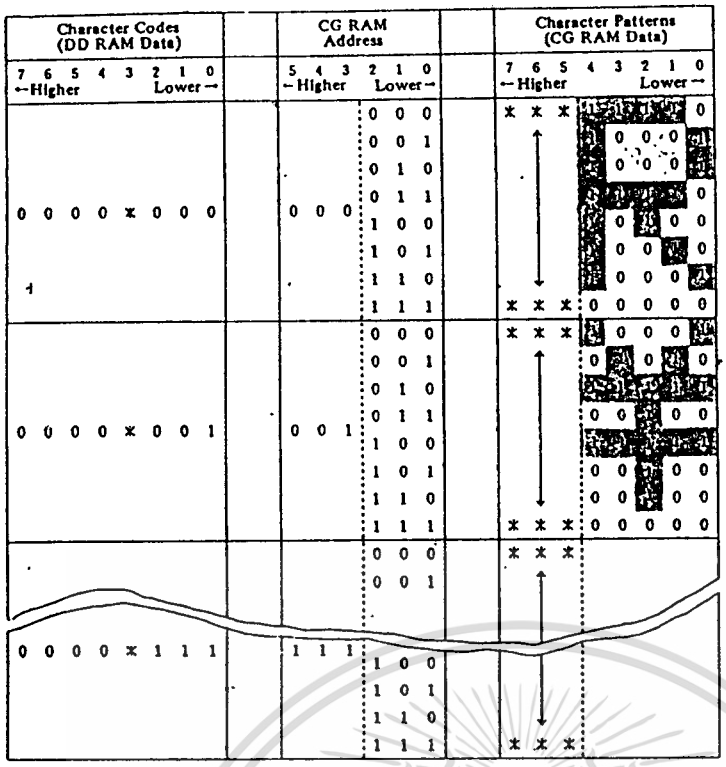
เป็นคำสั่งเขียนข้อมูลเข้าไปใน CG หรือ DD RAM โดยเมื่อเขียนข้อมูลและ ADDRESS จะเพิ่มหรือลดโดยอัตโนมัติตามคำสั่งที่ SET ใน ENTRY MODE ข้อกำหนดที่จะรู้ว่าเป็นการเขียนข้อมูลของ CG RAM หรือ DD RAM ทำได้โดยการ SET ADDRESS ของ CG RAM หรือ DD RAM ขึ้นมาก่อนจะเขียนข้อมูล

11. READ DATA FROM CG OR DD RAM

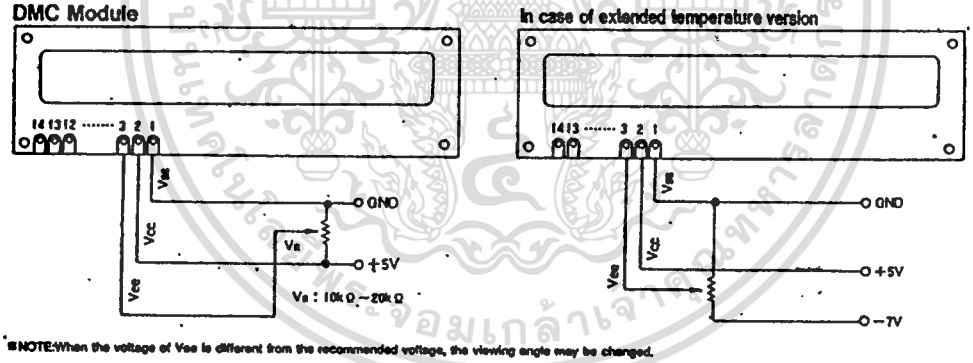
Read data from CG or DD RAM

	RS	R/W	DB_7						DB_0
Code	1	1	D	D	D	D	D	D	D	D

← Higher Order Bits Lower Order Bits →



สรุป การใช้งาน LCD MODULE นั้นที่สำคัญคือ ต้องเข้าใจในตัว CONTROLLER ของ LCD MODULE นั้น โดย CONTROLLER ทุกๆบริษัทจะมีการทำงานที่เหมือนๆกันเป็นส่วนใหญ่



CHARACTER FONT TABLE

NOTE: CGRAM is a CHARACTER GENERATOR RAM having a storage function of character pattern which enable to change freely by user's program.

บรรณานุกรม

- ไมโครโปรเซสเซอร์ Z-80 วิบูลย์ ชื่นแขก (KAITN)
- ทฤษฎีและการประยุกต์ไมโครโปรเซสเซอร์ Z-80 ยืน ภู่วรวรรณ
- Z-80 Assambly langquage programming. Berheley Calif. Osbome and Associates 1979.