ปีการศึกษา 2530

68000 MACHINE

โดย

นายนพดล หัทยานนท์

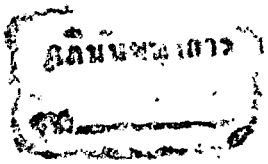
นายบุญชัย สินพารานนท์

นายไพโรจน์ อมรเวชยกุล

อาจารย์ที่ปรึกษา

รศ.มนัส สังวรศิลป์

อ.วัชระ ลัดรวิริยะ



ปริญญานิพนธ์ปีการศึกษา 2530

ภาควิชา วิศวกรรมอิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง 68000 MACHINE

ผู้จัดทำ

1.	นายนพดล	หัทยานนท์	271100	ภาควิชาอิเลคทรอนิคส์
2.	นายบุญชัย	สินพารานนท์	271104	ภาควิชาคอมพิวเตอร์
3.	นายไพโรจน์	อมรเวชยกุล	271147	ภาควิชาคอมพิวเตอร์

..... อาจารย์ที่ปรึกษา

(.....)

..... อาจารย์ที่ปรึกษา

(.....)

68000 MACHINE

บุญชัย สินพารานนท์
ไพโรจน์ อมรเวชยกุล
นพดล หัทย่านนท์
อ.วัชระ ฉัตรวิริยะ อาจารย์ที่ปรึกษา
รศ.มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2530

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เรียบเรียงขึ้น จากผลงานที่ได้ออกแบบและพัฒนาขึ้นเป็นระบบไมโครคอมพิวเตอร์ขนาดเล็กที่มี MC 68000 เป็นซีพียูบนเมนบอร์ด มีคีย์บอร์ดและจอมอนิเตอร์ของ IBM PC ทำหน้าที่เป็นอินพุทและเอาท์พุทตามลำดับ โดยทำการติดต่อกันผ่านทาง RS-232 และมีหน่วยความจำในขั้นต้น 8 กิโลไบท์ แบ่งเป็น EPROM 4 กิโลไบท์ และเป็น RAM 4 กิโลไบท์ จุดประสงค์เพื่อใช้ในการศึกษาและเป็นต้นแบบสำหรับการพัฒนาระบบไมโครคอมพิวเตอร์ 32 บิทในอนาคต

นอกจากนี้ภายในเมนบอร์ด จะมีโปรแกรมมอนิเตอร์ทำหน้าที่รับรหัส Machine (Machine Code) ซึ่งเป็นผลลัพธ์ที่ได้จากโปรแกรม 68000 Assembler ที่สร้างขึ้น มาทำการประมวลผล เสร็จแล้วจะส่งผลที่ได้ตอบกลับไปยัง IBM PC

68000 MACHINE

Bunchai Sinparanont
Pairote Amornvejayakul
Noppadol Hattayanont
Vatchara Chatviriya Advisor
Manus Sangvornsin Advisor
1987

Abstract

This thesis is an application of small sized Microcomputer System that MC 68000 is a CPU on mainboard. There are keyboard and CRT monitor of IBM PC as input and output in order. They are interfacing by RS-232. It includes 8 Kbyte memory, 4 Kbyte EPROM and 4 Kbyte RAM. The purpose is used for study and form for the Microcomputer system 32 bit development in the future.

The addition, the internal of mainboard has monitor program that receives machine code , is result from the 68000 Assembler Program which created for bring to process. Afterward it will send the results return to the IBM PC.

สารบัญ

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมาของโครงงาน

นับวันไมโครคอมพิวเตอร์ได้เข้ามามีบทบาทต่อสังคมไทยมากขึ้น นับตั้งแต่ไมโครคอมพิวเตอร์ขนาด 8 บิทอย่างแอปเปิ้ล (Apple) จนกระทั่งถึงไมโครคอมพิวเตอร์ขนาด 16 บิท และ 32 บิท อย่างเช่น ไอบีเอ็ม พีซี/เอ็กซ์ที/เอที (IBM PC/XT/AT) และ แมคอินทอช (Macintosh) เป็นต้น

ในปัจจุบันจะเห็นได้ว่า เครื่องไมโครคอมพิวเตอร์รุ่นใหม่ๆ ที่ออกมาจะยิ่งมีประสิทธิภาพสูงขึ้นเรื่อยๆ ทั้งนี้เพราะความก้าวหน้าทางเทคโนโลยี่ทำให้อุปกรณ์ต่างๆ มีขีดความสามารถสูงขึ้น ในบรรดาอุปกรณ์ต่างๆ ที่ประกอบขึ้นเป็นไมโครคอมพิวเตอร์นั้น ซีพียู(CPU) หรือไมโครโปรเซสเซอร์นับเป็นหัวใจสำคัญและมีบทบาทมากที่สุด ที่จะกำหนดว่าไมโครคอมพิวเตอร์เครื่องนั้น จะมีขีดความสามารถ หรือทำอะไรได้มากแค่ไหน

ไมโครโปรเซสเซอร์เบอร์หนึ่งที่กำลังได้รับความนิยมมาก ก็คือ MC 68000 ของบริษัทโมโตโรล่า เครื่องรุ่นใหม่ที่ใช้ MC 68000 เป็นซีพียูก็มีอาทิ เช่น แมคอินทอชของบริษัทแอปเปิ้ล, ซินแคลร์ โอแอล (Sinclair OL) ตลอดจนไอบีเอ็ม เอ็กซ์ที/370 เพอร์ซันนอลคอมพิวเตอร์ (IBM XT/370 Personal Computer) ของไอบีเอ็ม

แต่สำหรับในด้านสังคมไทยแล้ว ไมโครคอมพิวเตอร์ในระดับนี้ถึงแม้จะมีประสิทธิภาพสูง แต่ก็ยังมีราคาสูงด้วย ทำให้ในด้านของการศึกษาและการใช้งานจริง จึงไม่กว้างขวางเท่าที่ควร รวมทั้งบอร์ด 68000 ที่ต่างประเทศผลิตขึ้นมา ราคาก็ยังสูงเช่นเดียวกัน ดังนั้นจึงได้ทำโครงงานนี้ขึ้นเพื่อเป็นจุดเริ่มต้นในการแก้ปัญหาเรื่องนี้

1.2 วัตถุประสงค์ของโครงงาน

โครงงานนี้ได้ทำการสร้างและพัฒนาระบบซึ่งเป็นไมโครคอมพิวเตอร์แบบง่ายๆ โดยมี MC 68000 ทำหน้าที่เป็นซีพียู และมีคีย์บอร์ด (Key board) กับ จอภาพซีอาร์ที (CRT) ของ IBM PC ทำหน้าที่เป็นอินพุทกับเอาท์

พุทตามสาตับ รวมทั้งมีการรับส่งข้อมูลโดยผ่านอาร์เอส-232 (RS-232) ทั้งนี้เพื่อใช้ในการศึกษาคำสั่งและลักษณะการทำงานของ MC 68000 ว่ามีลักษณะแตกต่างจากไมโครโปรเซสเซอร์ตัวอื่นอย่างไรบ้าง และเพื่อใช้เป็นต้นแบบในการพัฒนาระบบ ไมโครคอมพิวเตอร์ 32 บิทต่อไป

1.3 วิธีการดำเนินงาน

จากโครงงานนี้เราได้ศึกษาระบบ โดยเริ่มต้นจากการทดลองวงจรในขั้นพื้นฐานก่อน ในลักษณะของฟรีรันโมด (Free Run Mode) เพื่อตรวจสอบดูว่าซีพียูที่ใช้สามารถใช้งานได้จริง รวมทั้งมีวงจรสร้างคล็อค (Clock Generator) และวงจรรีเซ็ท (Reset) ประกอบด้วย โดยทำการทดสอบวงจรย่อยต่างๆ ที่เกี่ยวข้องดังกล่าวทีละส่วนๆ ก่อนที่จะนำมาประกอบหรือต่อเข้าด้วยกันเป็นระบบ

หลังจากนั้นจึงทำการเพิ่มส่วนของวงจรย่อยอื่นๆ เช่นวงจรที่เกี่ยวข้องกับหน่วยความจำรอมและแรม (ROM and RAM) ของระบบ และวงจรย่อยอื่นๆ อีกบางส่วน ในลักษณะของระบบน้อยที่สุด (Minimum System) คือระบบที่สามารถใช้งานได้โดยมีอุปกรณ์น้อยที่สุดเท่าที่จำเป็น

ทางด้านซอฟท์แวร์ก็แบ่งออกเป็น 2 ส่วน คือ

1. ส่วนของโปรแกรมมอนิเตอร์ ซึ่งเขียนขึ้นจากภาษาแอสเซมบลี้ของ 68000 ประกอบด้วยคำสั่ง "D" (Dump) , "S" , "X" , "T" (Trace) , "G" (Go) โดยเชื่อม (link) กับคำสั่ง "A" (Assembler) , "U" (Disassembler) ที่เขียนขึ้นจากภาษาซี (C Language) บนเครื่องไอบีเอ็ม พีซี

2. ส่วนของโปรแกรม 68000 Assembler บนไอบีเอ็ม พีซี โดยทำการดัดแปลงจากโปรแกรม "A" จากลักษณะที่เก็บแบบพอยน์เตอร์มาเป็นแบบตาราง (Table)

1.4 ขอบเขตของโครงงาน

- ออกแบบและสร้างแผ่นบอร์ด 68000 ที่มีอาร์เอส-232 ในการรับส่งข้อมูล

- เขียนโปรแกรมมอนิเตอร์สำหรับบอร์ด 68000 นี้ โดยทำ

หน้าที่รับรหัสแม็ชชีน (Machine Code) จากไอบีเอ็ม พีซี มาทำการประมวลผล แล้วส่งผลลัพธ์ที่ได้กลับไปยังไอบีเอ็ม พีซี

- เขียนโปรแกรมแอสเซมเบลอร์ และดิสแอสเซมเบลอร์ (Assembler and Disassembler) ใน DEBUG

- เขียนโปรแกรมคอมไพเล่อร์ที่เป็น 68000 แอสเซมเบลอร์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

โครงงานนี้จะเป็นการเริ่มต้น สำหรับการศึกษาระบบไมโครคอมพิวเตอร์ในระดับ 16/32 บิท. โดยเฉพาะการศึกษาไมโครโปรเซสเซอร์ MC 68000 อุปกรณ์ที่สร้างขึ้นจะใช้ต้นทุนที่ถูกกว่า สามารถนำไปศึกษาหรือนำไปใช้งานได้ง่ายและสะดวกขึ้น ผู้ที่มีเครื่องไมโครคอมพิวเตอร์ไอบีเอ็ม พีซี อยู่แล้วสามารถที่จะนำบอร์ดนี้ไปใช้งานได้เลย โดยติดต่อผ่านทางอาร์เอส-232 สำหรับผู้ที่มีเครื่องไมโครคอมพิวเตอร์ยี่ห้ออื่นๆ ก็สามารถที่จะนำไปใช้งานได้เช่นเดียวกัน เพียงแต่เปลี่ยนโปรแกรมทางซอฟท์แวร์ที่ใช้งานบางส่วนเท่านั้น

บทที่ 2

ลักษณะการทำงานและคุณสมบัติของ MC 68000

2.1 รีจิสเตอร์

MC 68000 เป็นไมโครโปรเซสเซอร์ที่มีโครงสร้างทั้งแบบ 16 บิท และ 32 บิท โดยมี

- ดาต้ารีจิสเตอร์ (Data Register) ขนาด 32 บิท 8 ตัว
- แอดเดรสรีจิสเตอร์ (Address Register) ขนาด 32 บิท 7 ตัว
- สแตคพอยน์เตอร์ (Stack Pointer) ขนาด 32 บิท 2 ตัว
- โปรแกรมเคาน์เตอร์ (Program Counter) ขนาด 32 บิท แต่ 24 บิท ล่างเท่านั้นที่ทำหน้าที่ส่งสัญญาณออกมา
- และสเตตัสรีจิสเตอร์ (Status Register) ขนาด 16 บิท

2.1.1 ดาต้ารีจิสเตอร์ (D0-D7)

MC 68000 มีดาต้ารีจิสเตอร์ ขนาด 32 บิท แต่สามารถที่จะใช้งานได้แบบไบท์ (byte : 8 บิท) เวิร์ด (word : 16 บิท) หรือ ลองเวิร์ต (long word : 32 บิท) ก็ได้

ในการใช้งาน ดาต้ารีจิสเตอร์ ไม่ถึง 32 บิท บิทที่ไม่เกี่ยวข้องจะไม่มีการเปลี่ยนแปลง ซึ่งแสดงถึงการชิพท์ (Shift) แบบอริธเมติก (Arithmatic) ขนาด 8 บิท ดาต้ารีจิสเตอร์นี้ นอกจากจะใช้เป็นตัวรับส่งข้อมูลแล้ว ยังสามารถใช้เป็นอินเด็กซรีจิสเตอร์ (Index Register) ได้ด้วย

2.1.2 แอดเดรสรีจิสเตอร์ (A0-A6)

แอดเดรสรีจิสเตอร์ ขนาด 32 บิทนี้สามารถใช้งานแบบเวิร์ด

หรือ ลองเวิร์ด แต่ไม่สามารถใช้งานแบบไบท์ได้ ในการใช้รับข้อมูลขนาด 16 บิท บิทที่ 16-31 จะมีค่าตามไซจ์นบิท (Sign Bit : บิทที่ 15) ทั้งนี้จะมีประโยชน์ในการบวกหรือลบแอดเดรส

การที่ MC 68000 มีดาต้ารีจิสเตอร์ และแอดเดรสรีจิสเตอร์ขนาด 32 บิทนี้เป็นลักษณะพิเศษ คือไม่มีแอคคิวมูเลเตอร์รีจิสเตอร์ (Accumulator Register) ผิดกับไมโครโปรเซสเซอร์ 16 บิทตัวอื่นๆ เช่น 8086 , Z-8000 เพราะ MC 68000 สามารถใช้รีจิสเตอร์ตัวไหนเป็นแอคคิวมูเลเตอร์ก็ได้ ไม่มีการเจาะจงรีจิสเตอร์ตัวหนึ่งตัวใดโดยเฉพาะ วิธีการนี้ใช้ในเครื่องมินิคอมพิวเตอร์ หรือเครื่องเมนเฟรม อย่างเช่นเครื่องไอบีเอ็ม 370

2.1.3 สแตคพอยน์เตอร์ (A7)

เนื่องจาก MC 68000 สามารถทำงานได้ 2 โหมด (Mode) ด้วยกัน คือ ซุปเปอร์ไวเซอร์โหมด (Supervisor Mode) และ ยูสเซอร์โหมด (User Mode) ดังนั้น MC 68000 จึงมีสแตคพอยน์เตอร์ 2 ตัวด้วยกัน ตัวหนึ่งใช้ในซุปเปอร์ไวเซอร์โหมด ส่วนอีกตัวหนึ่งใช้ในยูสเซอร์โหมด แต่ในเวลาเดียวกัน MC 68000 สามารถทำงานได้เพียงโหมดเดียว จึงมีเพียงพอยน์เตอร์ตัวเดียวที่ทำงานในขณะหนึ่งๆ

2.1.4 โปรแกรมเคาน์เตอร์ (PC)

MC 68000 มีโปรแกรมเคาน์เตอร์ขนาด 32 บิท แต่ใช้เพียง 24 บิท ซึ่งจะใช้หน่วยความจำแอดเดรส (Address Memory) ได้ขนาด 16 เมกกะไบท์ ($=2^{24}$) เนื่องจาก MC 68000 เป็นไมโครโปรเซสเซอร์ขนาด 16 บิท ดังนั้นข้อมูลพื้นฐานจะมีขนาด 16 บิท แต่สามารถเรียกใช้แบบ 8 บิท หรือ 32 บิทก็ได้ ในการเรียกใช้ข้อมูลขนาด 16 บิท หรือ 32 บิทนี้ แอดเดรสจะต้องเป็นเลขคู่ ลักษณะเช่นนี้เป็นลักษณะทั่วไปของไมโครโปรเซสเซอร์ขนาด 16 บิท ซึ่งมีผลทำให้ค่าในโปรแกรมเคาน์เตอร์ และสแตคพอยน์เตอร์ต้องเป็นเลขคู่ (คือ บิทที่ 0 มีค่าเป็น "0") ตามไปด้วย

2.1.5 สเตตัสรีจิสเตอร์

MC 68000 ใช้สเตตัสรีจิสเตอร์ ขนาด 16 บิท ซึ่งแบ่งเป็น 2 ส่วน คือ ไบท์ของระบบ (System byte) กับ ไบท์ของผู้ใช้ (User byte)

- ไบท์ของผู้ใช้

บิทแคร์รี่ (Carry : C) ทำหน้าที่เหมือนกับ แคร์รี่ในไมโครโปรเซสเซอร์ขนาด 8 บิททั่วๆ ไป คือเป็น "1" เมื่อมีตัวทดจากการบวก หรือมีตัวยืมจากการลบ นอกจากนี้แล้วคำสั่งโรเต็ต (Rotate) และชิพท์ (Shift) ยังมีผลกับบิทๆ นี้

บิทโอเวอร์โฟลว์ (Overflow : V) บิทนี้ใช้ในการคำนวณเลขที่มีเครื่องหมาย คำที่ได้เกิดจากการเอ็กซ์คิวซิฝว์ ออร์ (Exclusive OR) ระหว่างแคร์รี่ แฟลก(Carry Flag) กับ ไซจ์นบิท การที่บิทนี้มีค่าเป็น "1" แสดงว่าขนาดคำตอบใหญ่เกินกว่าที่รีจิสเตอร์จะเก็บค่าไว้ได้

บิทซีโร่ (Zero : Z) บิทนี้จะมีค่าเป็น "1" เมื่อคำตอบมีค่าเป็น 0 เท่านั้น นอกจากนี้แล้วบิทนี้จะมีค่าเป็น "0"

บิทเนกาทิฝ (Negative : N) บิทนี้จะมีค่าตามไซจ์นบิทของคำตอบ ถ้าบิทนี้มีค่าเป็น "0" แสดงว่าคำตอบเป็นเลขบวก แต่ถ้าบิทนี้มีค่าเป็น "1" แสดงว่าคำตอบเป็นลบ

บิทเอ็กซ์เท็นด์ (Extend : X) บิทนี้ใช้เป็นแคร์รี่ ในการคำนวณที่ต้องการความเที่ยงตรงมากๆ (multipresision) คำสั่งที่มีผลกับบิทนี้อย่างเช่นคำสั่งบวก คำสั่งลบ คำสั่งชิพท์ บิทนี้จะมีค่าตามลิทแคร์รี่

สำหรับบิทที่เหลืออยู่ ในปัจจุบันยังไม่มีกวารใช้งาน และมีค่าเป็น "0" เสมอ

- ไบท์ของระบบ จะแสดงถึงสภาวะของระบบ ในขณะที่ไบท์ของผู้ใช้จะเป็นเงื่อนไขต่างๆ ซึ่งเป็นผลจากโปรแกรม บิทต่างๆ ในไบท์ของระบบนี้จะเปลี่ยนค่าได้ เมื่ออยู่ในซุปเปอร์ไวเซอร์โหมดเท่านั้น

เอส-บิท (S-bit) เป็นบิทที่แสดงว่า MC 68000 กำลังทำงานอยู่ในโหมดไหน ถ้าบิทนี้มีค่าเป็น "1" จะอยู่ในซุปเปอร์ไวเซอร์โหมด

และถ้าบิทนี้เป็น "0" ก็จะอยู่ในยูสเซอร์โหมด โหมดทั้ง 2 นี้จะมีสแตคพอยน์เตอร์ของตัวเอง และมีบางคำสั่งที่สามารถใช้ได้ในซุปเปอร์ไวเซอร์โหมดเท่านั้น โดยทั่วไปแล้วซุปเปอร์ไวเซอร์โหมดนี้ มักถูกใช้สำหรับระบบปฏิบัติการ (Operating System) ส่วนยูสเซอร์จะใช้สำหรับโปรแกรมใช้งานต่างๆ (Application Program)

ที-บิท (T-bit) เป็นบิทที่ใช้แสดงว่า ขณะนั้น MC 68000 อยู่ในเทรซโหมด (Trace Mode) หรือไม่ ซึ่งถ้าบิทนี้มีค่าเป็น "0" จะไม่อยู่ในเทรซโหมด แต่ถ้ามีค่าเป็น "1" ก็จะอยู่ในเทรซโหมด เทรซโหมดนี้จะมีลักษณะคล้ายกับการแทรพ (Trap คือ การอินเทอร์รัพท์ ชนิดหนึ่งที่เกิดจากซอฟท์แวร์ ไม่ใช่สัญญาณอินเทอร์รัพท์ที่ส่งเข้ามาจากอุปกรณ์ภายนอก) และทำให้กลับไปที่ซุปเปอร์ไวเซอร์โหมด โดยอาจจะมีโปรแกรมดีบักเกอร์ (Debugger) อยู่คอยแสดงผลของคำสั่งนั้นๆ

2.2 ขาและสัญญาณควบคุม

MC 68000 เป็นไอซีที่มีขนาด 64 ขา ประกอบด้วยสัญญาณต่างๆ ดังนี้

- ดาต้าบัส และแอดเดรสบัส (Data Bus & Address Bus)

เนื่องจาก MC 68000 เป็นไมโครโปรเซสเซอร์ขนาด 16 บิท ดังนั้นหน่วยของข้อมูลที่ใช้โดยทั่วไปจึงมีขนาด 16 บิท นั่นหมายความว่าในการรับส่งข้อมูลแต่ละครั้งจะรับส่งได้ไม่เกิน 16 บิท ถ้าจะรับส่งเกิน 16 บิท ก็ต้องทำการรับส่งมากกว่า 1 ครั้ง ข้อมูลที่รับส่งระหว่าง MC 68000 กับอุปกรณ์ภายนอก (อาจจะเป็นหน่วยความจำ หรือ อุปกรณ์อินพุท เอาท์พุทต่างๆ) จะรับส่งผ่านดาต้าบัส (D0-D15)

เพื่อที่จะให้อุปกรณ์ภายนอกทราบว่า MC 68000 จะติดต่อกับอุปกรณ์ไหน MC 68000 จะใช้สัญญาณจากแอดเดรสบัส (A1-A23) เลือกอุปกรณ์โดยส่งแอดเดรสของอุปกรณ์นั้นไปตามขา A1-A23 พร้อมกับมีสัญญาณแอดเดรสสโตรบ (Address Strobe : AS) เพื่อบอกอุปกรณ์ภายนอกว่า ตอนนี้แอดเดรส

, บนแอดเดรสบัส เป็นแอดเดรสที่ถูกต้องแล้ว

- การรับส่งข้อมูลแบบอะซิงโครนัส (Asynchronous)

เนื่องจาก MC 68000 สามารถเรียกใช้ข้อมูลขนาด 1 ไบท์ได้ ทำให้อ้างแอดเดรสได้ถึง 16 เมกกะไบท์ ซึ่งต้องใช้สัญญาณแอดเดรสขนาด 24 บิท แต่ว่า MC 68000 มีขาแอดเดรสเพียง 23 ขา ดังนั้นจึงมีสัญญาณ LDS (Lower Data Strobe) และ UDS (Upper Data Strobe) ช่วย โดยเมื่อสัญญาณ LDS เป็น "0" ข้อมูลจะถูกรับส่งในตาต้าบัส 8 บิทล่าง (D0-D7) และถ้าสัญญาณ UDS เป็น "0" ข้อมูลจะถูกรับส่งในดาต้าบัส 8 บิทบน (D0-D15) ดังนั้นถ้าสัญญาณทั้งสองเป็น "0" ข้อมูลทั้ง 16 บิทจะถูกรับส่งพร้อมกัน.

ต่อมาเพื่อที่จะให้อุปกรณ์ภายนอกทราบว่า MC 68000 กำลังจะรับ (Read) หรือส่ง (Write) MC 68000 ก็จะใช้สัญญาณ R/W เป็นตัวบอก คือ

ถ้าสัญญาณ R/W เป็น "1" แสดงว่า MC 68000 กำลังต้องการรับข้อมูล

และถ้าเป็น "0" แสดงว่ากำลังส่งข้อมูล การทำงานของสัญญาณ UDS, LDS และ R/W

หลังจากที่เราสามารถบอกได้ว่า อุปกรณ์ไหนที่ใช้รับส่งข้อมูลไบท์ไหน หรือทั้ง 2 ไบท์ และจะทำการรับหรือส่งข้อมูลแล้ว สิ่งต่อไปก็คือ การตอบรับของอุปกรณ์ภายนอกมายัง MC 68000 ว่าส่งข้อมูลมาให้แล้วหรือรับข้อมูลไว้แล้ว ซึ่งจะใช้สัญญาณดาต้าทรานซเฟอร์แอคโนว์เลดจ์ (Data Transfer Acknowledge : DATCK) ถ้าไม่มีสัญญาณนี้ตอบรับกลับมา MC 68000 จะรอไปเรื่อยๆ วิธีนี้จะทำให้ MC 68000 สามารถที่จะชลอการทำงาน เมื่อติดต่อกับอุปกรณ์ที่ทำงานช้าและทำงานได้เร็วขึ้น เมื่อติดต่อกับอุปกรณ์ที่ทำงานได้เร็ว จะเห็นได้ว่า การรับส่งข้อมูลแบบนี้จะอาศัยการตอบรับ (hand shaking) ไม่ใช่กำหนดช่วงเวลาให้รับส่งให้ทันการรับส่งข้อมูลโดยอาศัยการตอบรับนี้ การรับส่งเป็นแบบอะซิงโครนัส

- สัญญาณรหัสฟังก์ชั่น (Function Code)

MC 68000 นอกจากจะมีสัญญาณตอบรับสำหรับส่งข้อมูลแล้ว ยังมีสัญญาณที่ใช้บอกเหตุการณ์ภายนอกว่า ข้อมูลที่รับส่งนั้นเป็นข้อมูลอะไร เช่น โปรแกรม หรือ ข้อมูลของผู้ใช้ หรือ โปรแกรม หรือ ข้อมูลของซุปเปอร์ไวเซอร์ สัญญาณที่ใช้บอกนี้คือสัญญาณรหัสฟังก์ชั่น (FC0,FC1,FC2)

สัญญาณรหัสฟังก์ชั่นนี้ สามารถใช้ขยายเนื้อที่หน่วยความจำจาก 16 เมกกะไบท์เป็น64 เมกกะไบท์ได้ นอกจากนี้แล้วยังสามารถใช้สัญญาณรหัสฟังก์ชั่นจัดการเกี่ยวกับหน่วยความจำอย่างเช่น ไม่ให้โปรแกรมผู้ใช้ใช้เนื้อที่หน่วยความจำบางส่วนที่สงวนไว้สำหรับการทำงานในซุปเปอร์ไวเซอร์โหมด

- การรับส่งข้อมูลแบบซิงโครนัส (Synchronous)

นอกจากจะรับส่งข้อมูลแบบอะซิงโครนัส ซึ่งใช้การตอบรับโดยไม่ขึ้นกับความเร็วของอุปกรณ์ที่ติดต่อด้วยแล้ว MC 68000 ยังสามารถรับส่งข้อมูลแบบซิงโครนัส การรับส่งแบบซิงโครนัสนี้ MC 68000 ไม่ได้รอสัญญาณ DATCK แต่ใช้สัญญาณคล็อค (Clock) เพื่อกำหนดช่วงเวลาในการรับส่งแทน

การรับส่งแบบซิงโครนัส ใช้สัญญาณเอ็นเอเบิล (Enable : E) เป็นตัวส่งสัญญาณคล็อคดังกล่าว จาก MC 68000 โดยใช้ความถี่เพียง 1/10 ของสัญญาแคล็อค ที่ MC 68000 ใช้จริง และรูปสัญญาณจะมีขนาด 40/60 คือเป็น "1" อยู่ 4 คล็อค และเป็น "0" อยู่ 6 คล็อค

การรับส่งแบบซิงโครนัสนี้ อุปกรณ์ภายนอกจะส่งสัญญาณ VPA (Valid Peripheral Acknowledge) มาให้ MC 68000 เพื่อบอกให้เราทราบว่าต่อไปจะรับส่งข้อมูลแบบซิงโครนัส MC 68000 จะตอบรับทราบแล้วโดยใช้สัญญาณ VMA (Valid Memory Acknowledge)

การที่ MC 68000 สามารถที่จะรับส่งข้อมูลแบบซิงโครนัส ทำให้ MC 68000 สามารถใช้อุปกรณ์สนับสนุนที่เคยใช้กับ MC 68000 ได้ MC 6800 เป็นไมโครโปรเซสเซอร์ขนาด 8 บิทของโมโตโรลา เช่นเดียวกัน และจัดว่าเป็นไมโครโปรเซสเซอร์ตระกูลหนึ่งที่มีชื่อเสียง และมีอุปกรณ์ประกอบเป็นจำนวนมาก ซึ่งอุปกรณ์เหล่านั้นส่วนมากก็มีการรับส่งข้อมูลแบบซิงโครนัสด้วยวิธีนี้ MC 68000 ก็มีอุปกรณ์ประกอบเพิ่มขึ้นอีกมาก ซึ่งไมโครโปรเซสเซอร์ขนาด 16 บิทอื่นๆ ก็ใช้วิธีนี้อย่างเช่น 8086 ของอินเทล (Intel) สามารถใช้อุปกรณ์ร่วม

กับ 8080 อันเป็นไมโครโปรเซสเซอร์ 8 บิท ที่มีชื่อเสียงของบริษัทอินเทลเองได้

- สัญญาณควบคุมบัส และระบบ

สัญญาณ BR (Bus Request) , BG (Bus Grant) , BGACK (Bus Grant Acknowledge) เป็นสัญญาณควบคุมบัส สำหรับการทำ DMA (Direct Memory Access) สัญญาณ RESET , BERR (Bus Error) และ HALT เป็นสัญญาณ 2 ทิศทาง นอกจากอุปกรณ์ภายนอกจะส่งสัญญาณเข้ามาแล้ว MC 68000 ยังสามารถส่งสัญญาณนี้ไปควบคุมอุปกรณ์ภายนอกโดยตัว MC 68000 ไม่ต้องถูกฮอลท์ (Halt) หรือรีเซ็ท (Reset) จริงๆ อีกด้วย สัญญาณ BERR เป็นสัญญาณที่ใช้ประกอบกับ DTACK โดยเมื่ออุปกรณ์ภายนอกเกิดความผิดพลาด ไม่สามารถที่จะส่งสัญญาณ DTACK ได้ สัญญาณ BERR ก็จะถูกส่งเข้ามาบอก MC 68000 (อาจส่งมาจากอุปกรณ์อีกตัวหนึ่ง ซึ่งทำหน้าที่ตรวจสอบความผิดพลาด) ว่ามีข้อผิดพลาด (Error) เกิดขึ้น

- สัญญาณอินเทอร์รัพท์

ประกอบด้วยสัญญาณ 3 ตัว คือ สัญญาณ IP0, IP1 และ IP2 ซึ่งรวมกันแล้ว จะแสดงถึงระดับไพรออริตี้ หรือระดับความสำคัญของการอินเทอร์รัพท์ การที่สัญญาณอินเทอร์รัพท์จะทำการอินเทอร์รัพท์ได้นั้นจะต้องมีระดับไพรออริตี้สูงกว่า หรือเท่ากับที่มาส์คไว้ในสเตตัสรีจิสเตอร์ ถ้าสามารถอินเทอร์รัพท์เข้ามาได้ MC 68000 ก็จะไปประมวลผลรูทีน (Routine) ต่างๆ ตามเวคเตอร์ (Vector คือ แอดเดรสของรูทีนต่างๆ)

2.3 เอ็กซ์เซ็พท์ชั่น (Exception)

ก็คือการอินเทอร์รัพท์ ที่บริษัทโมโตโรล่า ใช้ชื่อนี้เพราะ MC 68000 สามารถทำการอินเทอร์รัพท์ได้มากชนิดกว่าไมโครโปรเซสเซอร์เบอร์อื่นมาก

วิธีการจัดการกับเอ็กซ์เซ็พท์ชั่นนั้น MC 68000 จะใช้ตาราง ซึ่งเก็บค่าเวคเตอร์ที่จะจัดการกับการอินเทอร์รัพท์ชนิดต่างๆ เมื่อ MC 68000 รู้ว่าเป็นอินเทอร์รัพท์ชนิดไหน ก็จะสามารถไปทำรูทีนนั้นตามค่าเวคเตอร์ในตาราง ตารางที่ใช้เก็บค่าเวคเตอร์นี้มีชื่อว่า ตารางเอ็กซ์เซ็พท์ชั่นเวคเตอร์

(Exception Vector Table) ไมโครโปรเซสเซอร์ขนาด 16 บิทชนิดอื่น เช่น 8086 และ 8000 ก็ใช้วิธีเดียวกันนี้ ต่างกันที่ว่า MC 68000 มีตาราง ที่ใหญ่กว่ามาก นอกจากนี้ยังสามารถแบ่งการอินเทอร์รัพท์จากภายนอกที่เข้ามา ได้ถึง 7 ระดับ

2.3.1 โหมดของการทำงานกับการเกิดเอ็กซ์เซ็พท์ชั่น

ดังที่กล่าวมาแล้ว MC 68000 แบ่งการทำงานออกเป็น 2 โหมด คือ ซุปเปอร์ไวเซอร์โหมด กับ ยูสเซอร์โหมด เมื่อ MC 68000 ถูกรีเซ็ท มันจะเริ่มทำงานในซุปเปอร์ไวเซอร์โหมดจนกว่าจะมีคำสั่งที่เปลี่ยนค่าบิท ในสเตตัสรีจิสเตอร์ให้เป็น "0" จึงจะไปทำงานในยูสเซอร์โหมดต่อไป

เมื่อ MC 68000 ทำงานอยู่ในยูสเซอร์ มีแต่การเอ็กซ์เซ็พท์ชั่นเท่านั้นที่จะทำให้ MC 68000 กลับไปทำงานในซุปเปอร์ไวเซอร์โหมดได้

2.3.2 ชนิดของการเอ็กซ์เซ็พท์ชั่น

แบ่งออกเป็น 2 ชนิดใหญ่ๆ ดังนี้คือ

1. การเอ็กซ์เซ็พท์ชั่นที่เกิดจากภายใน ซึ่งมีผลเกิดจากคำสั่งบางคำสั่ง เช่น คำสั่ง Trap (เป็นซอฟท์แวร์อินเทอร์รัพท์ เทียบได้กับคำสั่ง Restrat ของ 8080) หรือเกิดจากความผิดพลาดบางอย่างที่พบภายใน ได้แก่

- แอดเดรสผิด หรือ แอดเดรสเออร์เร่อร์ MC 68000 พยายามที่จะเรียกใช้ข้อมูลขนาด 16 บิท หรือ 32 บิท หรือคำสั่งที่มีแอดเดรสเป็นเลขคี่
- พยายามที่จะทำคำสั่งที่ไม่อนุญาตให้ใช้ในยูสเซอร์โหมด
- คำสั่งผิด หรือ ไม่มีในชุดคำสั่ง

MC 68000 นอกจากจะมีคำสั่ง Trap โดยตรงแล้ว ยังมีคำสั่งอื่นอีกที่จะทำให้เกิดการ Trap เมื่อมีเงื่อนไขสอดคล้องกับที่ต้องการ อย่างเช่น ให้ Trap เมื่อเกิดโอเวอร์โฟลว์ (Overflow) ได้แก่ คำสั่ง TRAPV

นอกจากนี้อาจเกิดจากฟังก์ชั่นเทรซ (Trace) โดยที่ฟังก์ชั่นเทรซจะเริ่มทำเมื่อ ที-บิท ในสเตตัสรีจิสเตอร์เป็น "1" โดยหลังจากทำคำสั่งหนึ่งในยูสเซอร์โหมดแล้ว จะเกิดการเอ็กซ์เซ็พท์ชั่นขึ้นทันที ช่วยให้สามารถทำซิงเกิลสเตป (Single Step) ได้

2. การเอ็กซ์เซ็พท์ชั่นที่เกิดจากภายนอก ได้แก่สัญญาณที่อุปกรณ์ภายนอกส่งเข้ามาอินเทอร์รัพท์ คือ

- สัญญาณ BERR
- สัญญาณ RESET
- สัญญาณอินเทอร์รัพท์ (IPO, IP1, IP2)

2.3.3 ระดับความสำคัญของการเอ็กซ์เซ็พท์ชั่น

เมื่อเกิดการเอ็กซ์เซ็พท์ชั่นขึ้นหลายอย่างพร้อมๆ กัน MC 68000 จะจัดการกับเอ็กซ์เซ็พท์ชั่นที่มีระดับสูงสุดก่อน และในระหว่างการจัดการนั้นจะยอมให้มีการเอ็กซ์เซ็พท์ชั่นที่สูงกว่ามาเอ็กซ์เซ็พท์ชั่นได้เท่านั้น ระดับความสำคัญของการเอ็กซ์เซ็พท์ชั่นสามารถแบ่งออกเป็นชุดใหญ่ๆ ได้ทั้งหมด 3 ชุด คือ

- ชุดที่มีระดับความสำคัญสูงที่สุด จะหยุดการทำคำสั่งทันทีเมื่อเกิดการเอ็กซ์เซ็พท์ชั่นในชุดนี้ขึ้น โดยไม่ต้องรอให้ทำคำสั่งนั้นเสร็จ
- ชุดที่มีระดับความสำคัญรองลงมา จะรอให้ทำจนจบคำสั่งก่อน จึงเริ่มทำการเอ็กซ์เซ็พท์ชั่น ในชุดนี้จะรวมถึงการอินเทอร์รัพท์ ซึ่งแบ่งระดับความสำคัญออกอีก 7 ระดับด้วย
- ชุดที่มีระดับความสำคัญต่ำที่สุด จะรอให้จบคำสั่งก่อนเช่นเดียวกับชุดที่ 2

2.3.4 ตารางเอ็กซ์เซ็พท์ชั่นเวคเตอร์

ศูนย์กลางของการจัดการเอ็กซ์เซ็พท์ชั่นอยู่ที่ตารางนี้ ซึ่งมีขนาด 1024 ไบท์อยู่ในตำแหน่งหน่วยความจำที่ 000000H ถึง 0003FFH

ตารางนี้จะประกอบด้วยแอดเดรสขนาด 32 บิท (เพื่อที่จะโหลดเข้าโปรแกรมเคาน์เตอร์โดยตรง) จำนวน 256 แอดเดรส โดยที่เอ็กซ์เซ็พท์ชั่นแต่ละชนิดจะมีแอดเดรสที่แน่นอนที่ใช้เก็บแอดเดรสของรูทีนที่ใช้จัดการเอ็กซ์เซ็พท์ชั่นนั้นๆ ในตารางยังแบ่งเป็น 2 ส่วน คือ

- ส่วนที่กำหนดไว้แล้ว สำหรับเอ็กซ์เซ็พท์ชั่นชนิดต่างๆ ซึ่ง

ได้แก่ 64 แอดเดรส (เวคเตอร์) แรก

- และส่วนที่เหลืออีก 192 แอดเดรส (เวคเตอร์) นั้นผู้ใช้สามารถกำหนดได้เอง

นอกจากนี้แล้ว จะเห็นได้ว่ามีบางแอดเดรสที่ถูกสำรองไว้โดยบริษัทโมโตโรลาผู้ผลิตเอง

อุปกรณ์ภายนอกสามารถใช้แอดเดรสในเวคเตอร์ของผู้ใช้ได้ โดยเมื่อส่งสัญญาณอินเทอร์รัพท์เข้ามาแล้ว MC 68000 จะตอบโดยส่งอินเทอร์รัพท์แอคโนว์เลดจ์ (ใช้สัญญาณรหัสฟังก์ชั่น และมีสัญญาณ A1-A3 แสดงระดับความสำคัญที่อินเทอร์รัพท์เข้ามา) อุปกรณ์ก็จะส่งสัญญาณ DTACK พร้อมกับเลขที่เวคเตอร์มาตามสาย D0-D7 แต่ถ้าไม่ต้องการกำหนดเลขที่เวคเตอร์เองก็ส่งสัญญาณ VPA มา MC 68000 ก็จะทำการประมวลผลรูทีนตามที่กำหนดไว้แล้วสำหรับอินเทอร์รัพท์ระดับต่างๆ การที่ผู้ใช้สามารถกำหนดเวคเตอร์ได้เองนี้ ทำให้ MC 68000 สามารถทำการเอ็กซ์เซ็พท์ชั่นได้มากกว่า 200 ชนิด

2.4 โหมดในการแอดเดรส (Addressing Mode)

คำสั่งของ MC 68000 ส่วนใหญ่จะมีโอเปอร์แรนด์ (Operand) ตั้งแต่ 1 ตัวขึ้นไป โดยที่โอเปอร์แรนด์ตัวหนึ่งๆ อาจจะอยู่ภายในซีพียู (คือในรีจิสเตอร์) หรือภายนอก ซีพียู (คือในหน่วยความจำ)

โหมดในการแอดเดรสจะเป็นตัวกำหนดว่า ซีพียูจะจัดการกับแอดเดรสที่ใช้ในการประมวลผล (Effective Address) ของโอเปอร์แรนด์ที่อยู่ในรีจิสเตอร์หรือหน่วยความจำอย่างไร ซึ่ง MC 68000 มีวิธีการแอดเดรสได้ทั้งหมด 6 ชนิด คือ

1) การแอดเดรสแบบรีจิสเตอร์ไดเรคท์ (Register Direct Addressing) เป็นการกำหนดรีจิสเตอร์ที่จะใช้งานโดยตรง ได้แก่

- ดาต้ารีจิสเตอร์ไดเร็คท์ (Data Register Direct)

โหมดในการแอดเดรสโหมดนี้ ดาต้ารีจิสเตอร์จะเป็นโอเปอร์แรนด์ ซึ่งระบุโดยใช้นีโมนิค "Dn" โดยที่ "D" หมายถึง โอเปอร์แรนด์ที่เป็นดาต้ารีจิสเตอร์ และ "n" หมายถึง หมายเลขรีจิสเตอร์ตั้งแต่ 0 ถึง

- แอดเดรสรีจิสเตอร์ไดเร็คท์ (Address Register Direct)

คล้ายกับดาต้ารีจิสเตอร์ไดเร็คท์ ยกเว้นว่า รีจิสเตอร์ที่ใช้เป็นแอดเดรสรีจิสเตอร์ เราระบุโหมดนี้โดยใช้นีโมนิค "An" โดยที่ "A" หมายถึง แอดเดรสรีจิสเตอร์ และ "n" หมายถึง หมายเลขรีจิสเตอร์ตั้งแต่ 0 ถึง 7

2) การแอดเดรสแบบรีจิสเตอร์อินไดเรคท์ (Register Indirect Addressing) วิธีการแอดเดรสชนิดนี้ ค่าของแอดเดรสที่ต้องการจะอยู่ในแอดเดรสรีจิสเตอร์ ซึ่งอาจจะมีการบวก หรือลบด้วยค่าคงที่ หรือค่าในอินเด็กซ์รีจิสเตอร์ (Index Register จะเป็นดาต้ารีจิสเตอร์ หรือ แอดเดรสรีจิสเตอร์ก็ได้) ได้แก่

- รีจิสเตอร์อินไดเร็คท์ (Register Indirect)

การแอดเดรสชนิดนี้ โอเปอร์แรนด์จะอยู่ในหน่วยความจำ แอดเดรส รีจิสเตอร์ จะเก็บแอดเดรสของโอเปอร์แรนด์ไว้ โดยการระบุแอดเดรสรีจิสเตอร์ภายในวงเล็บเช่น (A3)

- พรีดีครีเมนท์ (Predecrement) รีจิสเตอร์อินไดเร็คท์

การแอดเดรสโหมดนี้ แอดเดรสรีจิสเตอร์จะเก็บค่าแอดเดรสไว้ในหน่วยความจำ อย่างไรก็ตามก่อนการกำหนดค่าแอดเดรสของโอเปอร์แรนด์ ซีพียูจะลบค่าออกจากแอดเดรสรีจิสเตอร์ คือค่าแอดเดรสที่อยู่ในหน่วยความจำ ค่าของการลบจะขึ้นอยู่กับขนาดของการโอเปอเรชั่น (Operation) คือ ลบด้วย 1 สำหรับไบท์โอเปอเรชั่น , 2 สำหรับเวิร์ดโอเปอเรชั่น .และ 4 สำหรับลองเวิร์ดโอเปอเรชั่น

หลังจากการลบ ซีพียูจะเก็บค่าใหม่ที่ได้นี้ไว้ในแอดเดรสรีจิสเตอร์ และใช้ค่านี้เป็นแอดเดรสที่ใช้ในการประมวลผลของโอเปอร์แรนด์ในแอสเซมเบลอสเตทเมนท์ (Assembler Statement) การแอดเดรสโหมดนี้จะระบุเครื่องหมาย "-" ก่อนวงเล็บ เช่น -(A5)

- โพสท์อินครีเมนท์ (Postincrement) รีจิสเตอร์อินได

เร็คท์

ในการแอดเดรสโหมดนี้คล้ายกับพรีดีครีเมนท์โหมด แอดเดรสรีจิสเตอร์จะมีแอดเดรสอยู่ในหน่วยความจำ และซีพียูจะเปลี่ยนแปลงค่าแอดเดรสรีจิสเตอร์ตามขนาดของการโอเปอเรชั่น

อย่างไรก็ตาม ในกรณีนี้ซีพียูจะใช้ค่าที่มีอยู่ในแอดเดรสรีจิสเตอร์ขณะนั้นเป็นแอดเดรสที่ใช้ในการประมวลผลของโอเปอร์แรนด์ หลังจากเก็บค่านี้แล้ว ซีพียูจะเพิ่มขนาดของการโอเปอเรชั่นเข้าไว้ในแอดเดรสรีจิสเตอร์ รูปแบบของการแอสเซมเบลอ (Assembler syntax) สำหรับโหมดนี้จะมีแอดเดรสอยู่ภายในวงเล็บ และตามด้วยเครื่องหมาย "+" เช่น (A5)+

- รีจิสเตอร์อินไดเร็คท์กับดิสเพลซเมนท์ (Register Indirect with Displacement)

ในการแอดเดรสโหมดนี้ แอดเดรสที่ใช้ในการประมวลผลของโอเปอร์แรนด์ จะเป็นผลบวกของดิสเพลซเมนท์ 16 บิทที่มีเครื่องหมายกับค่าที่อยู่ในแอดเดรสรีจิสเตอร์ ก่อนที่ซีพียูจะบวกค่าดิสเพลซเมนท์ กับค่าในแอดเดรสรีจิสเตอร์ มันจะทำการขยายเครื่องหมาย (sign extend) ของดิสเพลซเมนท์ จากบิทที่ 15 ไปเป็นบิทที่ 16 ถึง 31 ซึ่งจะมีทั้งดิสเพลซเมนท์ที่เป็นบวกและลบ (Positive & Negative)

ในโหมดนี้จะคล้ายกับรีจิสเตอร์อินไดเร็คท์โหมดอื่น รูปแบบของการแอสเซมเบลอจะใช้แอดเดรสรีจิสเตอร์อยู่ภายในวงเล็บ ค่าดิสเพลซเมนท์จะอยู่ก่อนแอดเดรสรีจิสเตอร์ เช่น 10(A1) โดยที่ค่าดิสเพลซเมนท์นี้จะเป็นค่าคงที่ ในขณะที่ค่าของแอดเดรส อาจเปลี่ยนไปตามการประมวลของโปรแกรม

3. การแอดเดรสแบบอิมไพลด์รีจิสเตอร์ (Implied Register Addressing) หมายถึงการใช้สแตคพอยน์เตอร์ , สเตตัสรีจิสเตอร์ หรือ โปรแกรมเคาน์เตอร์ ซึ่งจำเป็นต้องใช้อยู่แล้วสำหรับคำสั่งนั้นๆ

4) แอบโซลูทแอดเดรสซิ่ง (Absolute Addressing) เป็นการกำหนดแอดเดรสโดยตรงในคำสั่ง โดยที่แอดเดรสอาจจะเป็น 16 บิท หรือ 32 บิท ถ้าเป็น 16 บิทแล้วซีพียูจะทำการขยายเครื่องหมายก่อนใช้มัน

รูปแบบของการแอสเซมเบลอสำหรับโหมดนี้ ค่าแอดเดรสจะตามด้วย .L หรือ .W ถ้ามันรู้ค่าของแอดเดรสจะสามารถตัดสินใจได้ว่า แอดเดรสจะเป็น 16 หรือ 32 บิท

5) การแอดเดรสแบบโปรแกรมเคาน์เตอร์รีเลทีฟ (Program Counter Relative Addressing) เป็นการแอดเดรสโดยเทียบกับโปรแกรมเคาน์เตอร์ โดยจะทำการบวกค่าในโปรแกรมเคาน์เตอร์ ด้วยค่าในอินเด็กซ์รีจิสเตอร์ หรือ ค่าคงที่ ได้แก่

- โปรแกรมเคาน์เตอร์อินไดเร็คท์กับดิสเพลซเมนท์

ในโหมดการแอดเดรสนี้คล้ายกับโหมดของแอดเดรัสรีจิสเตอร์อินไดเร็คท์กับดิสเพลซเมนท์ ยกเว้นว่า ค่าแอดเดรสที่ใช้ในการประมวลผลจะเป็นค่าดิสเพลซเมนท์จากค่าที่อยู่ในโปรแกรมเคาน์เตอร์ขณะนั้นแทน ค่าของโปรแกรมเคาน์เตอร์ที่ใช้จะเป็นแอดเดรสของโอเปอร์แรนด์ในส่วนของคำสั่งนั้น ค่าดิสเพลซเมนท์ในโหมดนี้จะเป็น 16 บิทที่คิดเครื่องหมายด้วย โดยจะปิดดิสเพลซเมนท์และพีซี (PC : Program Counter) นิโมนิคไว้ในวงเล็บ เช่น 10(PC)

6) อิมมิเดียทดาต้าแอดเดรสซิ่ง (Immediate Data Addressing) เป็นการกำหนดข้อมูลโดยตรง อาจจะอยู่ในออพโค้ด (Opcode) เลย หรืออยู่ต่อจากออพโค้ด ก็ได้ โดยระบุตัวอักษร "#" ไว้หน้าโอเปอร์แรนด์ เช่น #123 และอาจต่อท้ายโอเปอร์แรนด์ด้วยตัวบอกขนาด คือ .B .W .L ถ้าหากไม่ใส่ ตัว Assembler จะเลือกขนาดตามขนาดของค่าเอง

2.5 ชุดคำสั่งของ 68000

MC68000 มีคำสั่งพื้นฐานอยู่ 56 คำสั่งด้วยกัน แต่เมื่อรวมกับการแอดเดรสอีกหลายชนิด ทำให้ชุดคำสั่งมีมากกว่า 300 คำสั่ง เนื่องจาก MC 68000 ไม่มีคำสั่งเฉพาะสำหรับอุปกรณ์อินพุท/เอาท์พุท แต่คำสั่งทุกคำสั่งที่อ้างถึงหน่วยความจำ ก็ใช้เป็นคำสั่งอินพุท/เอาท์พุทได้ เพราะ MC 68000 จะถือว่า อุปกรณ์อินพุท/เอาท์พุทเสมือนหน่วยความจำตำแหน่งหนึ่งที่อ่านหรือเขียนข้อมูลได้เช่นเดียวกับหน่วนความจำทั่วไป ซึ่งวิธีการนี้เรียกว่า "Memory Mapped

I/O"

รูปแบบของคำสั่งของ MC 68000 จะเริ่มด้วย Opcode ขนาด 2 ไบท์เสมอ ส่วนคำสั่งทั้งหมดอาจจะยาวได้ตั้งแต่ 2 ไบท์ถึง 10 ไบท์

คำสั่งส่วนใหญ่ที่อ้างถึงหน่วยความจำสามารถที่จะอ้างขนาด 1 ไบท์, เวิร์ด หรือ ลองเวิร์ด โดยในคำสั่งจะมี .B .W และ .L ต่อท้ายตามลำดับ

MC 68000 ไม่มีคำสั่งที่ใช้ย้ายข้อมูลเป็นบล็อค (Block) อย่างของ Z 8000 หรือ 8086 แต่ MC 68000 มีคำสั่งพื้นฐานที่ทำให้เขียนโปรแกรมที่ย้ายข้อมูลเป็นบล็อคได้ง่ายๆ

MC 68000 มีคำสั่งสำหรับการคูณ และการหาร ซึ่ง Z 8000 ก็มี โดย Z 8000 สามารถคูณ และหารข้อมูลได้เฉพาะแบบที่ไม่มีเครื่องหมายเท่านั้น ส่วน MC 68000 สามารถคูณ และหารแบบมีหรือไม่มีเครื่องหมายก็ได้ แต่ Z 8000 สามารถคูณและหารขนาด 32 บิทได้ ส่วน MC 68000 คูณและหารได้ แค่แบบ 16 บิทเท่านั้น

คำสั่งบางคำสั่งที่น่าสนใจของ MC 68000 ก็คือ คำสั่ง LINK และ UNLK คำสั่งคู่นี้จะใช้จองที่สำหรับจัดการเกี่ยวกับการส่งผ่านข้อมูลระหว่างโปรแกรมย่อย (Subroutine) ต่างๆ

บทที่ 3

การพัฒนาทางด้านฮาร์ดแวร์ (Hardware Development)

3.1. การสร้างระบบไมโครโปรเซสเซอร์

ระบบไมโครโปรเซสเซอร์ หมายถึงระบบที่ประกอบด้วยตัวไมโครโปรเซสเซอร์เป็นหลัก การทำงานจะเน้นที่ตัวไมโครโปรเซสเซอร์ ความแตกต่างระหว่างระบบไมโครโปรเซสเซอร์กับระบบไมโครคอมพิวเตอร์ ก็คือ ในระบบไมโครคอมพิวเตอร์จะประกอบด้วยส่วนต่างๆ 3 ส่วน คือ

1. ไมโครโปรเซสเซอร์ ซึ่งมีหน้าที่ประมวลผล
2. หน่วยความจำ มีหน้าที่เก็บข้อมูลและเก็บโปรแกรมสำหรับการทำงาน
3. ส่วนติดต่อกับอุปกรณ์ภายนอกอินพุท/เอาท์พุท มีหน้าที่แปลงสัญญาณต่างๆ จุดประสงค์เพื่อให้คอมพิวเตอร์สามารถติดต่อกับอุปกรณ์ภายนอกได้

ดังนั้นระบบไมโครโปรเซสเซอร์ ตามความหมายในที่นี้ก็คือระบบที่ประกอบด้วยซีพียูอย่างเดียวล้วนๆ

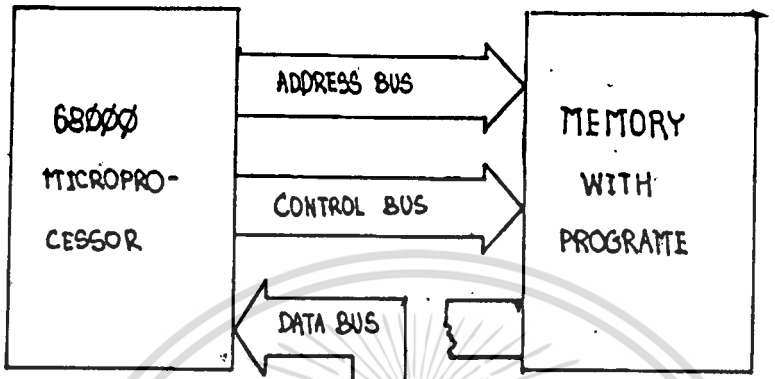
การพิจารณาระบบไมโครคอมพิวเตอร์ ก็ควรเริ่มจากระบบไมโครโปรเซสเซอร์ก่อน การที่จะให้ระบบไมโครโปรเซสเซอร์นี้ทำงานก็ควรจะให้มันทำคำสั่งที่ไม่ต้องทำอะไรเลย นั่นก็คือทำคำสั่ง NIL ซึ่งเป็นคำสั่งที่มีความยาว 16 บิท หรือ 1 เวิร์ด โดยตัวซีพียูจะทำคำสั่งนี้ซ้ำไปเรื่อยๆ จนครบ 16 เมกกะไบท์แล้วจะวนกลับมาทำอีก วิธีการที่จะให้ซีพียูทำคำสั่ง NIL ตลอด ก็คือ ตัดขาดาต้าออก แล้วนำคำสั่ง NIL ใส่เข้าไปแทน ดังในรูปที่ 3.1.

วิธีการนี้เรียกว่า วิธีฟรีรัน (Free Run) ซึ่งจะมีประโยชน์มากในการตรวจสอบระบบ โดยสามารถตรวจสอยการทำงานของสัญญาณต่างๆ เพราะถือว่า ขณะนี้ระบบกำลังทำงานอยู่จริง การตรวจสอบสัญญาณทำได้โดยใช้ออสซิโลสโคป หรือ ลอจิกโพรบ ขาที่ตรวจสอบได้ก็คือ ขาแอดเดรส และขาควบคุมต่างๆ ส่วนขาที่ยังไม่สามรถตรวจสอบได้ก็คือ ขาดาต้านั่นเอง

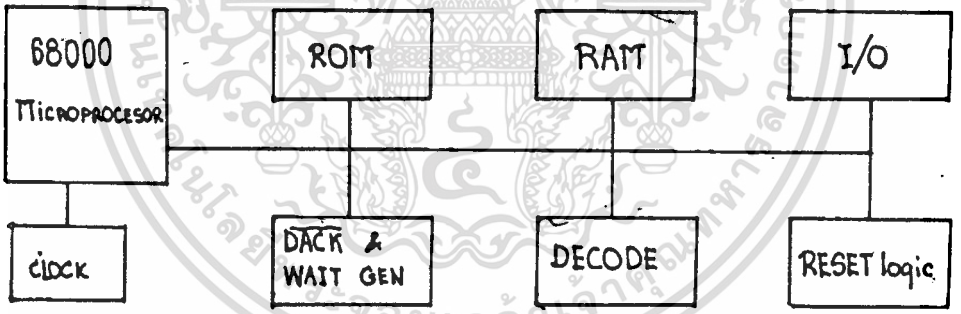
3.2 การออกแบบระบบขนาดเล็ก (MINIMUM SYSTEM)

ระบบขนาดเล็กในที่นี้หมายถึง ระบบไมโครโปรเซสเซอร์ ซึ่ง

มีรายละเอียดแยกเป็นบล็อกๆ ได้ดังรูปที่ 3.2



รูปที่ 3.1



รูปที่ 3.2

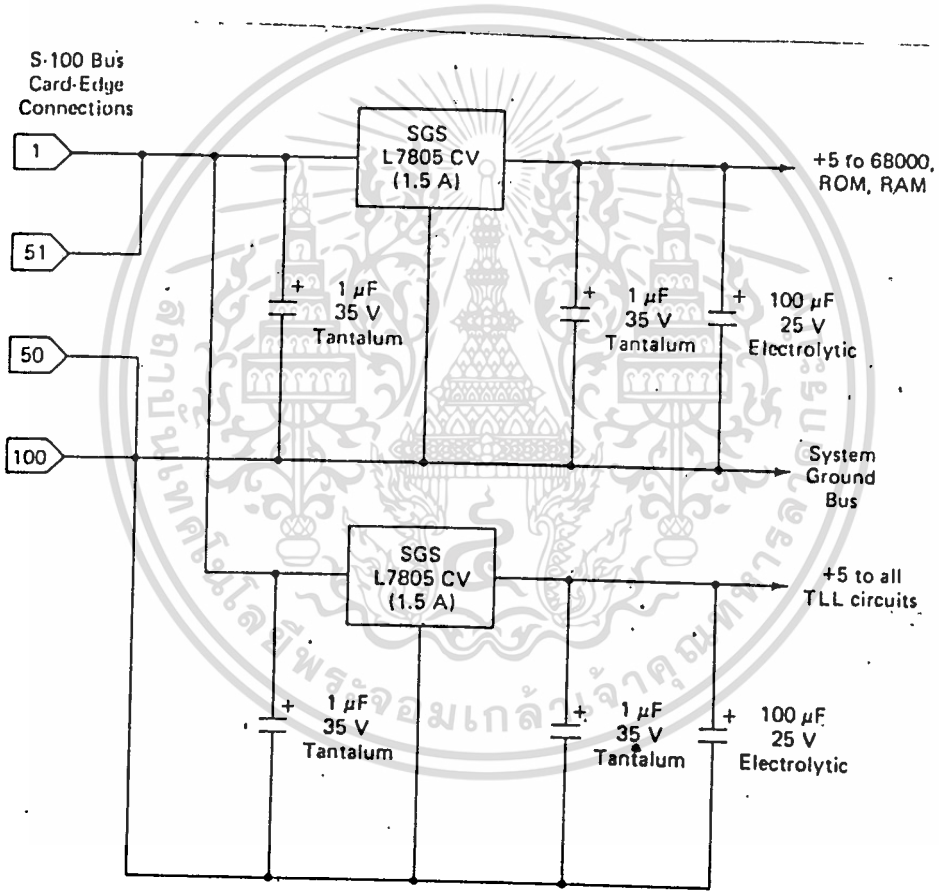
และเพื่อให้เป็นการง่ายต่อการพิจารณา เราจะมาพิจารณารายละเอียดของระบบภายในแต่ละบล็อก และถ้าแต่ละบล็อกสามารถทำงานตามขั้นตอนของมันแล้ว ก็จะสามารถประกอบขึ้นเป็นระบบใหญ่ได้

1. แหล่งจ่ายไฟ (Power Supply)

ในการออกแบบระบบแหล่งจ่ายไฟ มันเป็นการยากที่จะประมาณกระแสความต้องการของวงจรรวมทั้งหมด เพราะการออกแบบในขั้นตอนแรกยังกำหนดแน่นอนไม่ได้ว่า ต้องมีไอซีอะไรบ้าง และมีกี่ตัว

แต่ถ้าประมาณอย่างหยาบๆ โดยคิดว่ามีไอซีประมาณ 50 ตัว

แต่ละตัวมีกระแส 10 มิลลิแอมป์ (ใช้ LS-TTL) ก็จะได้ 0.5 แอมป์ ที่ 5 โวลท์ เฉพาะตัว 68000 ต้องใช้พลังงานโดยปรกติ 1.75 วัตต์สูงสุด หรือ 350 มิลลิแอมป์ที่ 5 โวลท์ แต่ในหนังสือข้อมูลของ 68000 บอกว่า มันอาจสามารถมียอดสูงสุดถึง 1.5 แอมป์ แต่เป็นการเปลี่ยนแปลงช่วงเวลาอันสั้นจากรูปที่ 3.3 จะใช้ 7805 ซึ่งจ่ายกระแสได้ 1.5 แอมป์ เป็นตัวปรับระดับโวลท์เตจ (regular)



รูปที่ 3.3

ที่ใช้ไอซี 7805 จำนวน 2 ตัวเพราะว่า การสวิง (swing) ของโวลท์เตจที่ TTL อาจรบกวนการทำงานของซีพียูได้

2. สัญญาณนาฬิกา (Clock)

MC 68000 มีความสามารถรัน (Run) ที่ความเร็วประมาณ 2,4,6 หรือ 8 เมกกะเฮิร์ตซ ตามสเป็คของแต่ละตัว สำหรับให้เลือกใช้งาน

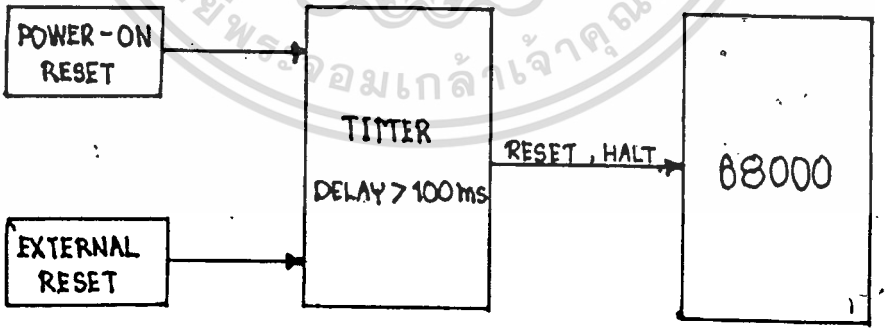
ซึ่งความถี่เหล่านี้ได้มาจากตัวคริสตอลนำมาต่อกันเป็นวงจรออสซิเลเตอร์ แต่การใช้งานสามารถเลือกใช้งานจากตัวออสซิเลเตอร์ได้เลย และเป็นการใช้งานที่สะดวกมาก ข้อควรระวังอย่างหนึ่ง คือ ตัวออสซิเลเตอร์ไม่สามารถมีกำลังจ่ายกระแสได้สูงมากนัก ดังนั้นจึงมีตัวขับกระแส (Driver) จำพวก 7400 หรือ 7404 แต่ก็ไม่สามารถต่อตรงจากออสซิเลเตอร์ได้เช่นกัน ต้องผ่านฟลิปฟลอบก่อน เพื่อให้เป็นรูปคล็อคที่ใช้งานได้ แต่จะทำให้ลดความถี่ของออสซิเลเตอร์ไปครึ่งหนึ่ง ดังนั้นอาจใช้ 74265 ซึ่งนำมาใช้งาน จะได้ความถี่ตามที่กำหนดไว้บนกระป๋อง

3. วงจรรีเซ็ท (Reset)

วงจรรีเซ็ทในระบบโดยทั่วไป จะทำหน้าที่ 2 ลักษณะคือ

- รีเซ็ทเมื่อเปิดเครื่อง (Power On)
- รีเซ็ทเมื่อกดปุ่มรีเซ็ท

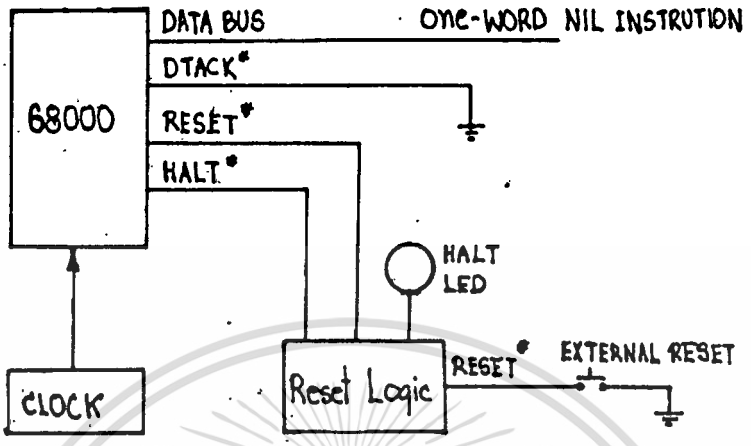
และโดยทั่วไป วงจรรีเซ็ทจะเป็นวงจรหน่วงเวลา จึงประกอบด้วยวงจรหน่วงเวลา R-C ดังรูปที่ 3.4 ซึ่งแสดงหน้าที่และลักษณะของวงจรรีเซ็ท



รูปที่ 3.4

4. ส่วนของไมโครโปรเซสเซอร์

จากรายละเอียดของวงจรดังที่กล่าวมาแล้วข้างต้นนั้นสามารถนำมารวมกัน เพื่อเขียนให้ดูง่ายขึ้น ดังรูปที่ 3.5



รูปที่ 3.5

จากที่กล่าวไปแล้ว เราต้องการให้ระบบอยู่ในสถานะฟรีรัน ซึ่งตามความหมายก็คือซีพียูจะเฟทช (Fetch) คำสั่ง NIL ซึ่งมีความยาว 1 เวิร์ด ในทุกๆ แอดเดรส คู่ต่อไป ทำเช่นนี้ไปเรื่อยๆ จนครบ 16 เมกกะไบท์ และเราสามารถตรวจสอบได้โดยวัดการเปลี่ยนแปลงสัญญาณที่ขาควบคุม (Control Lines) และขาแอดเดรส (Address Lines)

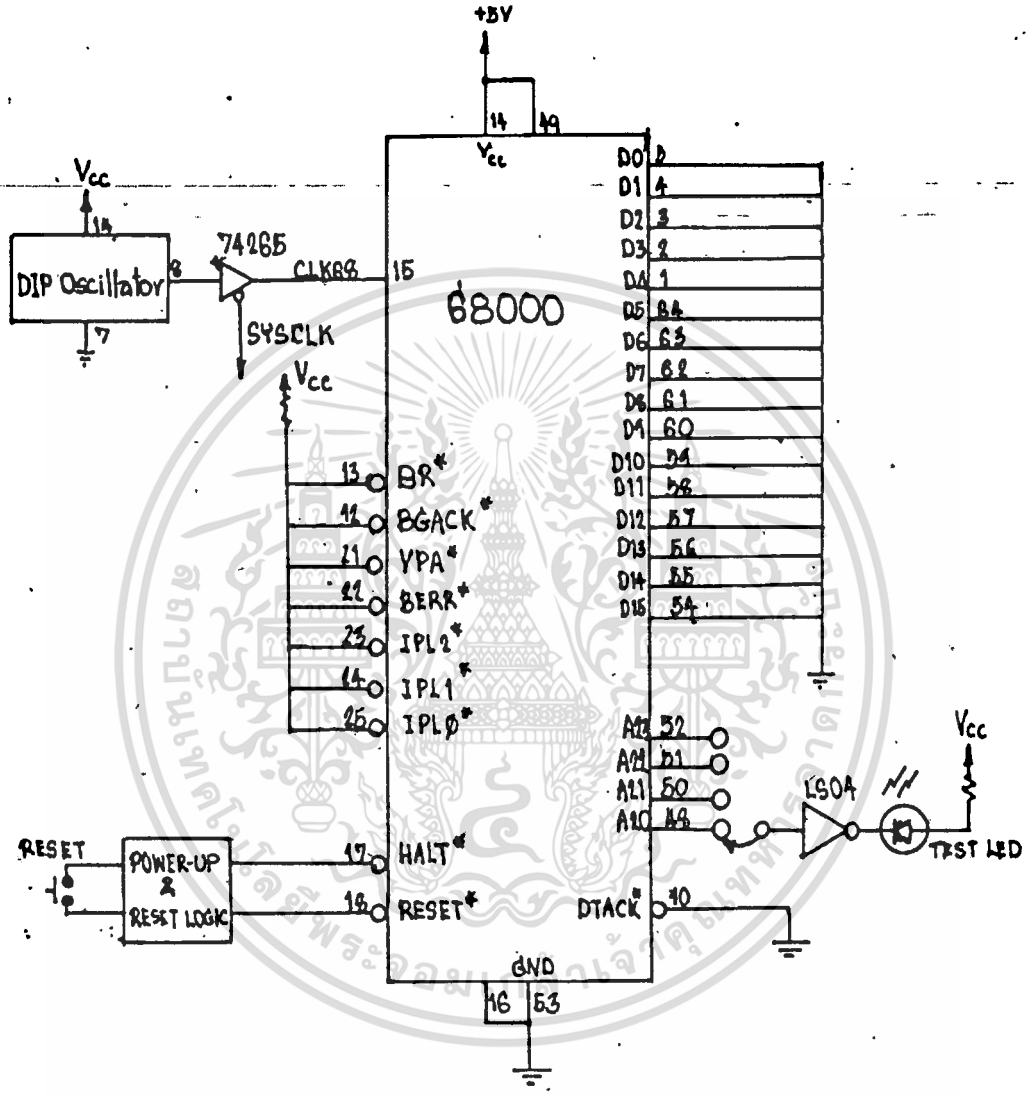
จากรูปที่ 3.6 แสดงวงจรจริงของซีพียูรวมกับวงจรอื่นๆ ที่กล่าวมาแล้ว ขาดาต้าทั้งหมดจะถูกต่อลงกราวด์ (Ground) ชั่วคราว เมื่อระบบถูกรีเซ็ทให้เริ่มทำงาน SSP ถูกเซ็ทเป็น "0" PC ถูกเซ็ทเป็น "0" ซึ่งเริ่มเฟทชคำสั่งที่มีออพโค้ด 0 และในความเป็นจริง ออพโค้ด 0 ก็ไม่ใช่คำสั่ง NOP คำสั่ง NOP จริงๆ แล้วมีออพโค้ด = 4E71H แต่ถ้าต่อให้ขาดาต้าเป็น 4E71 แล้วระบบฟรีรันจะไม่ทำงาน ดังนั้นเราจึงเรียก ออพโค้ด 00 เป็นคำสั่ง NIL ซึ่งการทำงานคล้ายๆ กับคำสั่ง NOP

พิจารณาออพโค้ด 0000 ซึ่งเป็นชุดคำสั่งจริงของ MC 68000 เมื่อมีความหมายเป็นภาษานีโมนิค (Mnemonic) ว่า ORI.B #0,D0 และมันถูกเลือกให้เป็นฟรีรันด้วยเหตุผล 2 ประการคือ

ประการแรก เพราะเป็นชุดคำสั่งที่เป็นเลขคู่

และประการที่สอง เพราะเป็นการแน่นอนกว่าที่จะต่อขาดาต้า

ลงกราวด์ทั้งหมด ตกว่าที่จะให้ขาใดขาหนึ่งเป็น "1"



รูปที่ 3.6

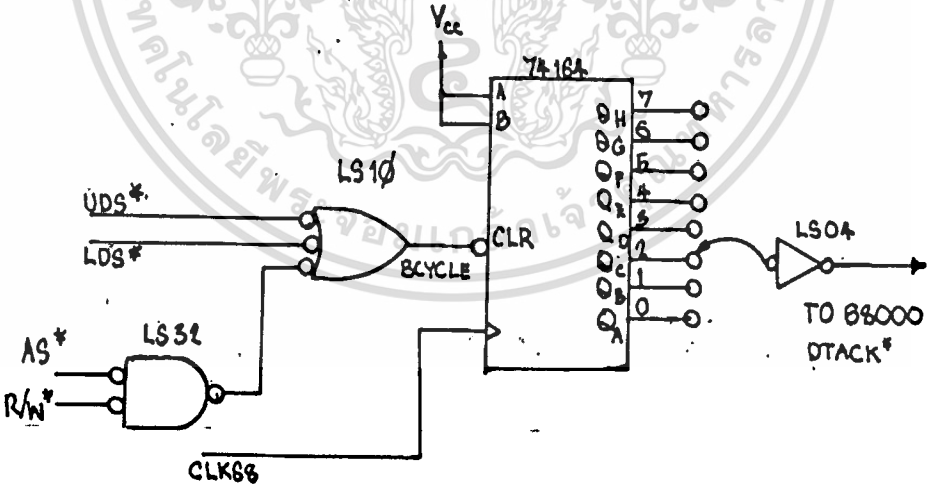
หน่วยความจำซึ่งมองเห็นได้จากตัวซีพียู MC 68000 จะเป็นลักษณะดังนี้

แอดเดรส	ดาต้า	โปรแกรม
000000	0000 0000	ORI.B #0,D0
000004	0000 0000	ORI.B #0,D0
000008	0000 0000	ORI.B #0,D0

00000C	0000 0000	ORI.B #0,D0
:	:	:
:	:	:
FFFFFC	0000 0000	ORI.B #0,D0

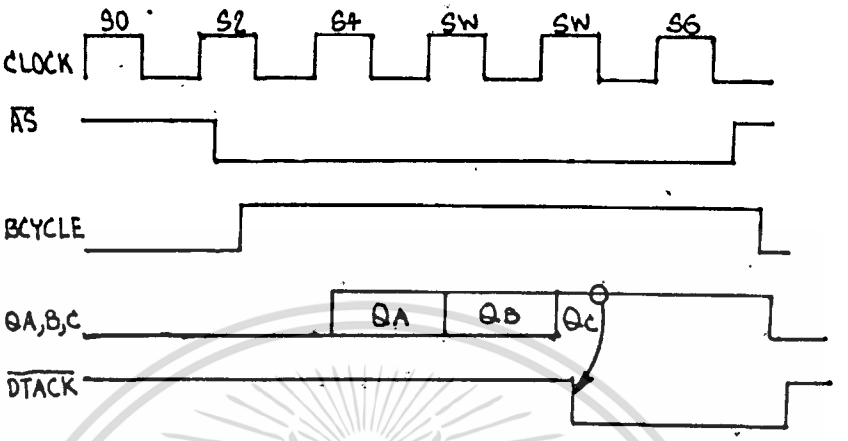
5. DTACK และ WAIT เจนเนอเรเตอร์ (DTACK and WAIT Generator)

ปรกติสัญญาณ DTACK จะเป็น low ตั้งแต่คล็อคที่ S4 ในแต่ละไซเคิล (Cycle) และจะกลับสู่สถานะเดิม หลังจาก AS เป็น high ถ้า DTACK ไม่เป็น low ที่ S4 แล้ว MC 68000 จะสร้างสถานะ WAIT ขึ้น ในกรณีที่ต่อ DTACK เป็น low ตลอด (ลงกราวด์) จะทำให้โปรเซสเซอร์ทำงานโดยไม่มีสถานะ WAIT วงจร DTACK และ WAIT เจนเนอเรเตอร์ แสดงดังรูปที่ 3.7



รูปที่ 3.7

จากรูปจะเห็นได้ว่า ถ้าในสภาวะปกติ DTACK จะเป็น high แต่หลังจากนั้นบัสไซเคิล (Bus cycle) จะเริ่มทำงานไประยะหนึ่งแล้ว DTACK จะถูกทำให้เป็น low จนกระทั่งหมดไซเคิล ดังรูป 3.8



รูปที่ 3.8

6. การออกแบบหน่วยความจำ (Memory Design)

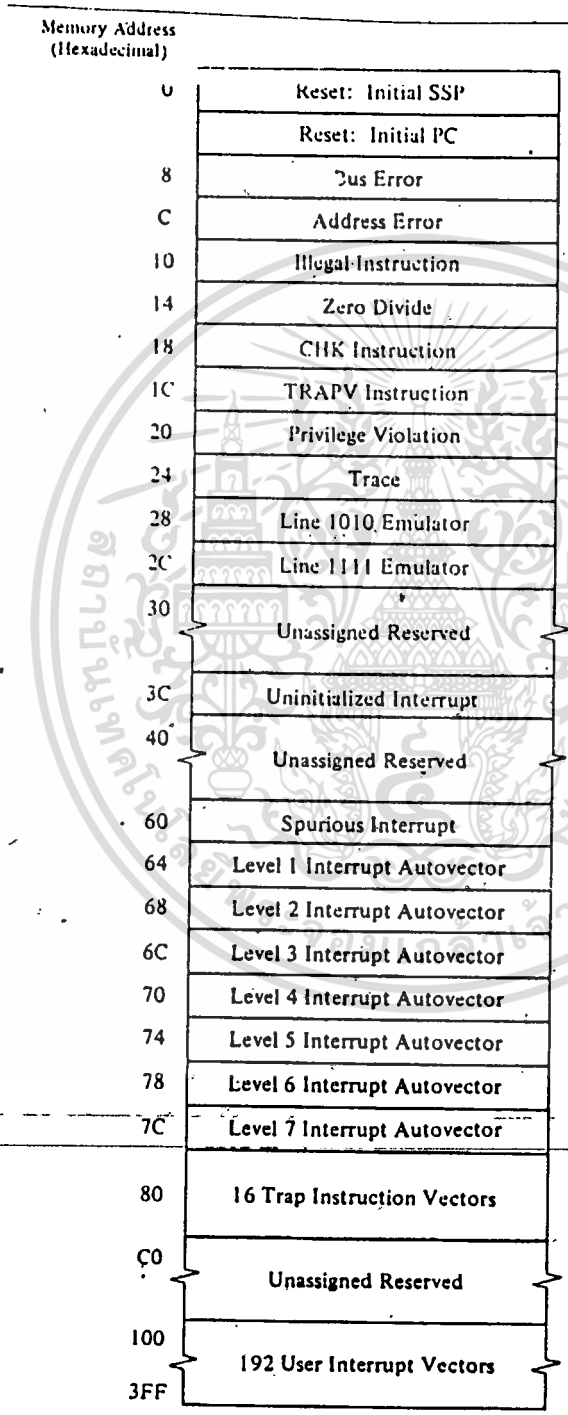
โดยปรกติแล้ว หลังจากรีเซ็ทระบบ MC 68000 จะนำค่าที่แอดเดรส 0 จำนวน 4 ไบท์ มาเป็นค่าของ SSP และค่าที่แอดเดรส 4 มา 4 ไบท์มาเป็นค่าของ PC เพื่อเป็นอินิเชียลพอยน์เตอร์ (initial pointer) ของโปรแกรมมอนิเตอร์ (Monitor Program) และมีตารางเอ็กซ์เซ็พท์ชั่นเวคเตอร์ เรียงต่อกันมาดังรูปที่ 3.9

เนื่องจากค่าของ SSP และ PC จะต้องคงอยู่เสมอ ดังนั้นจึงเก็บค่าเหล่านี้ไว้ในอีพรอม (EPROM) แล้วจึงเริ่มทำงาน โดยเปรียบเสมือนว่าค่าใน SSP และ PC ถูกเฟทชที่แอดเดรส 0000H แต่ตามความเป็นจริงแล้ว มันถูกเก็บไว้ที่แอดเดรส 8000H หลังจากที่บูทเครื่องแล้ว ก็จะถูกอ่าน SSP และ PC ที่แอดเดรสนี้ แล้วจึงไปทำงานตามที่ PC กำหนดไว้ ซึ่งอาจจะเป็นการโหลดเอ็กซ์เซ็พท์ชั่นเวคเตอร์ลงสู่แรม กำหนด I/O พอร์ต (I/O Port) หรืออะไรก็ได้ ในที่นี้สมมติให้ทำงานเพียงคำสั่งเดียว แล้ววนกลับมาใหม่ ดังรูป 3.10

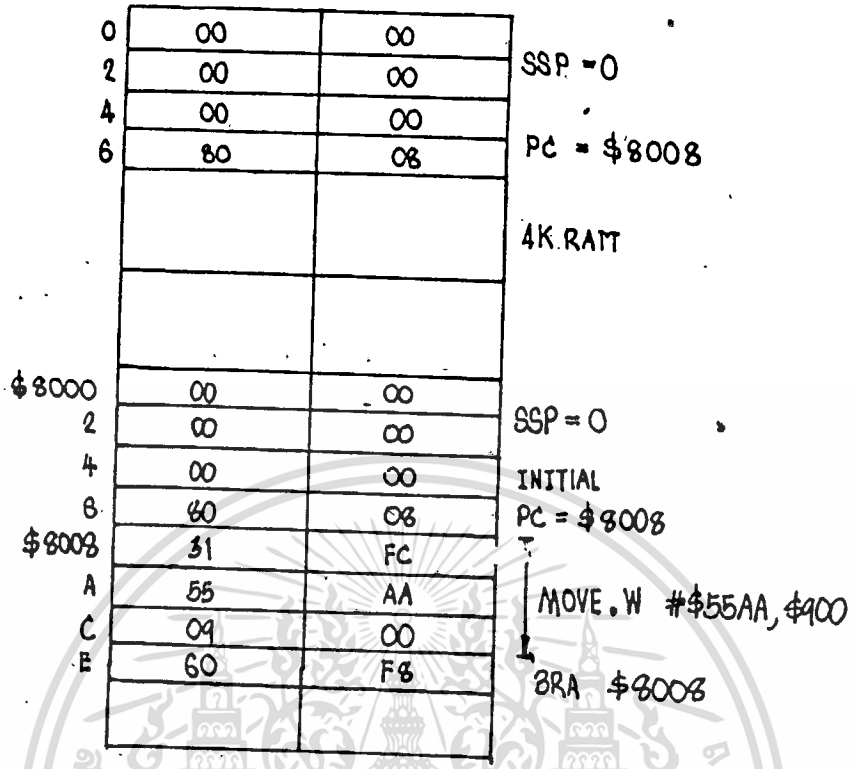
7. การเชื่อมต่อแบบชิงโครนัส (Synchronous Interface)

จะใช้ในการส่งผ่านข้อมูลระหว่างบอร์ด 68000 กับไอบีเอ็ม

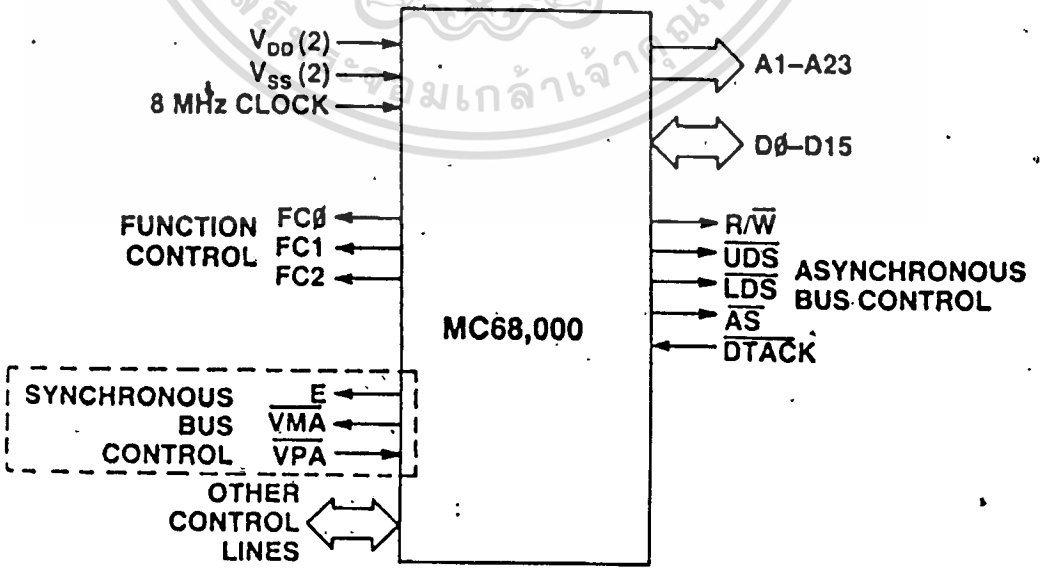
นิซ โดยใช้ส่วนของการควบคุมบัสแบบซิงโครนัส (Synchronous Bus Control) ที่อยู่ใน 68000 ในการควบคุม มีลักษณะดังรูป 3.11



รูปที่ 3.9



รูปที่ 3.10



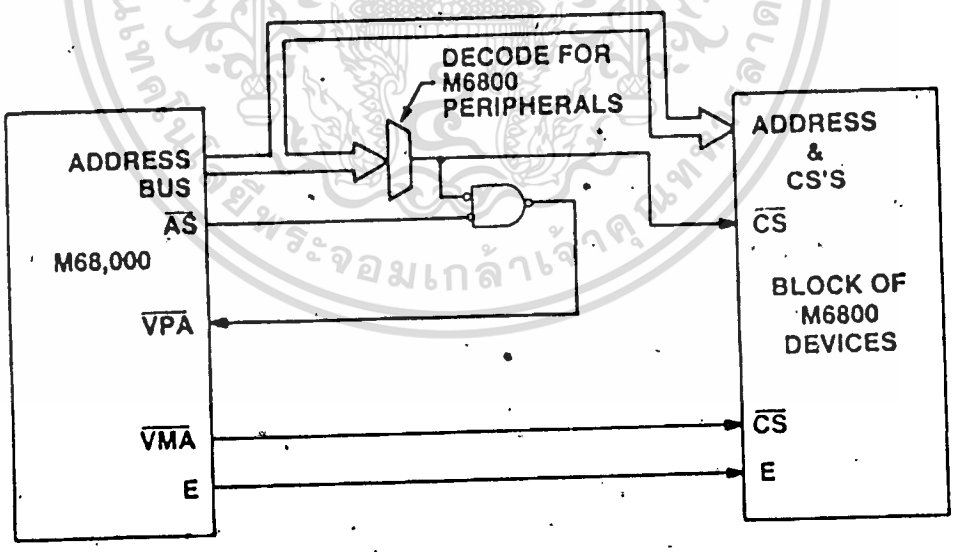
รูปที่ 3.11

- เมื่อบัสไซเคิลเริ่มต้น 68000 ก็จะทำการรอสัญญาณ DTACK

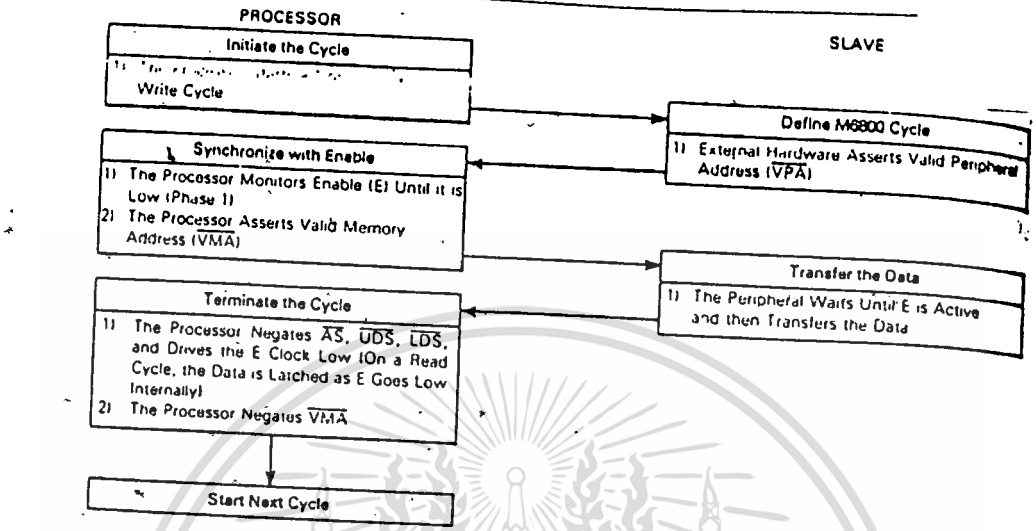
- ถ้าสัญญาณ VPA ถูกส่งมาแทนสัญญาณ DTACK แสดงว่าเริ่มต้นบัสไซเคิลแบบซิงโครนัส 68000 จะรอ สัญญาณ E เป็น low แล้วจึงส่งสัญญาณ VMA ออกไป

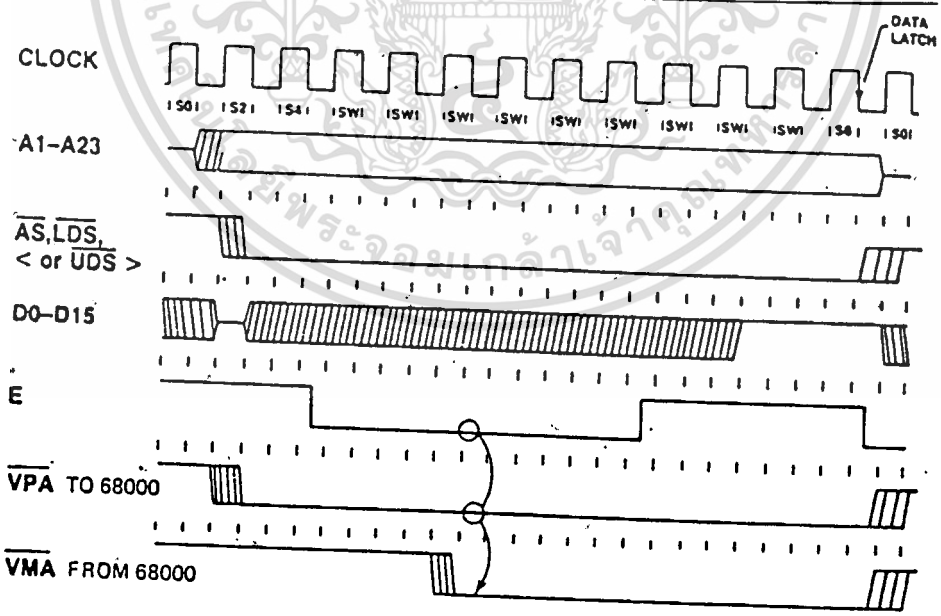
- เมื่อสัญญาณ E เป็น high ข้อมูลจะถูกแล็ทช์ (Latch) ลงสู่ตาต้าบัส รอจนสัญญาณ E เป็น low ข้อมูลก็จะถูกแล็ทช์สู่จุดเป้าหมายเป็นอันสิ้นสุดไซเคิล ดังรูป 3.12

ในที่นี้จะให้สัญญาณ VPA เกิดจากสัญญาณ AS กับแอดเดรสที่ถอดรหัส (Decode) ได้ หลังจากนั้น 68000 จึงส่งสัญญาณ VMA ออกมาเพื่อเป็นสัญญาณเลือกชิพ (Chip Select) ดังรูปที่ 3.13



รูปที่ 3.13





รูปที่ 3.12

3.3 วงจรที่ใช้งานจริงสำหรับโครงงานนี้

เนื่องจากระบบที่ออกแบบมาแล้วนั้น เพื่อจะได้เป็นต้นแบบให้กับการพัฒนาระบบไมโครคอมพิวเตอร์ ที่มี 68000 เป็นซีพียูต่อไป ดังนั้นระบบที่ออกแบบจะเน้นที่ความแน่นอนของระบบเสียก่อน จึงทำให้การออกแบบวงจรไม่ยุ่งยากลี้ลับซับซ้อนมากนัก โดย

- ในส่วนวงจรคล็อค จะใช้ออสซิเลเตอร์ 24 M ผ่านวงจรหาร 6 เพื่อให้เหลือ 4 M

- ในส่วนของรอม จะใช้เก็บค่าของ SSP และ PC ไว้ที่ 8 แอดเดรสแรก โดยมีแอดเดรสเริ่มต้นที่ 1000H และใช้ไอซี 2716 จำนวน 2 ตัว เป็นหน่วยความจำจำนวน 4 กิโลไบท์

- ในส่วนของแรม จะเริ่มที่แอดเดรส 0000H ใช้ไอซี 6116 จำนวน 2 ตัว เป็นหน่วยความจำจำนวน 4 กิโลไบท์

- ในส่วนของรีเซ็ท จะให้วงจร R-C ไทม์คอนสะแต็นท์ (R-C time constant) หน่วงเวลา โดยมีไอซี 7414 ช่วยปรับระดับ

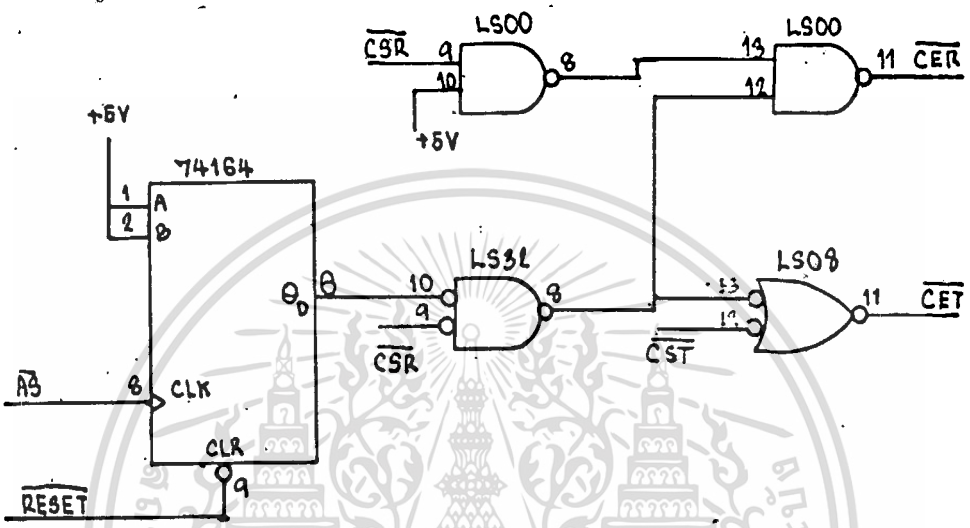
- ในส่วน DTACK จะใช้ดี-ฟลิปฟลอป (D-Flipflop) ในการหน่วงเวลา โดยมีสัญญาณ LDS,UDS,AS,R/W และแอดเดรสที่ถอดรหัสแล้วเป็นตัวเคลียร์ (Clear)

- ในส่วนของการถอดรหัส จะใช้ไอซี 74138 เพื่อถอดรหัสแรมที่แอดเดรส 0000H ถึง 07FFH , รอมที่แอดเดรส 1000H ถึง 17FFH และ I/O ที่แอดเดรส 0FF00H เป็นต้นไป

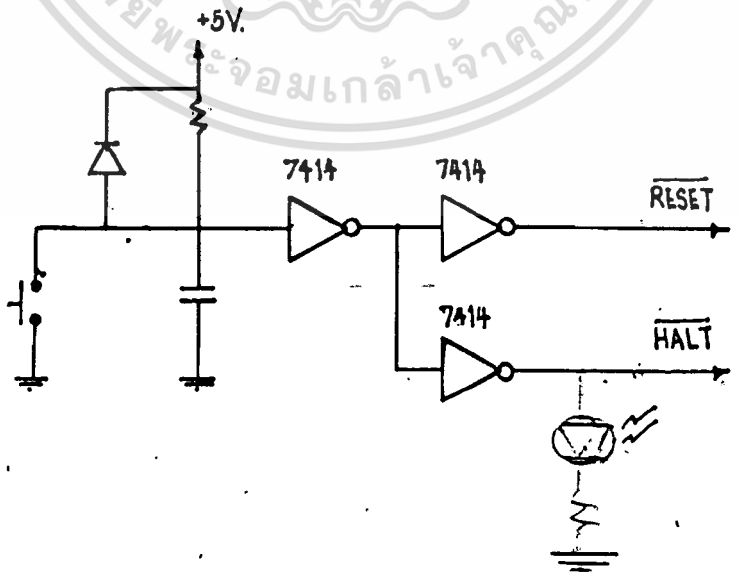
- ในส่วนบูทโมดูล (Boot Module) จะใช้ไอซี 74164 เพื่อนับสัญญาณ AS ที่เข้ามา 4 ครั้ง แสดงว่าค่า SSP และ PC ถูกเฟทช์ไปเรียบร้อยแล้ว จึงถอนตัวออกจากการบังคับการทำงานที่รอม เพื่อให้ทำงานตามโปรแกรมต่อไป

- ในส่วน I/O จะใช้ไอซี 8251 เป็นตัวรับส่งข้อมูลโดยจำลองการควบคุมบัสแบบซิงโครนัสในตระกูล 68XX มาใช้กับไอซี 8251

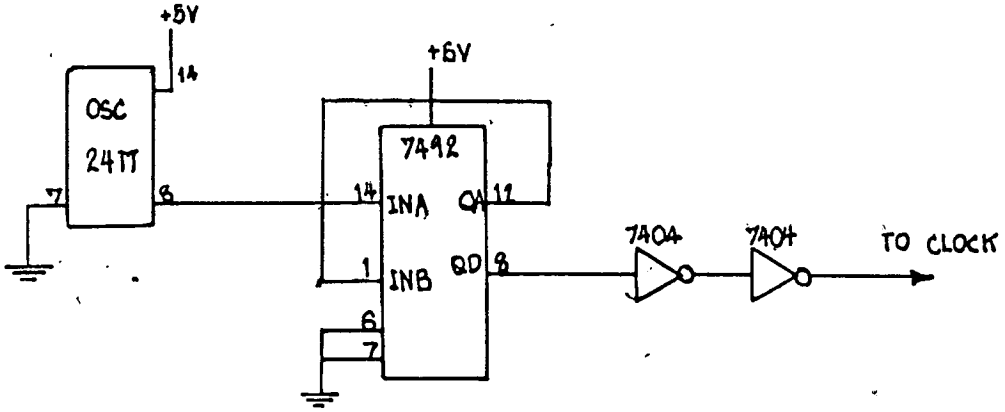
ต่อไปจะเป็นรูปของวงจรที่ใช้งาน



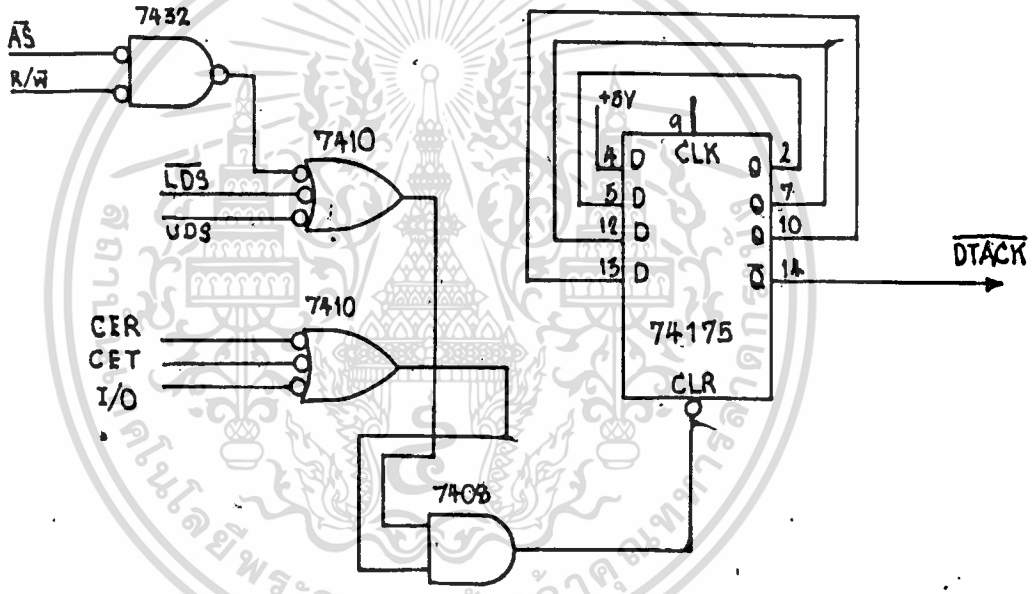
บูทโมดูล (Boot Module)



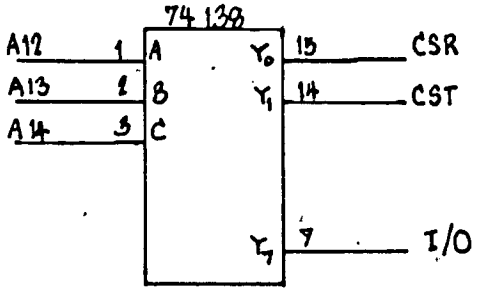
รีเซ็ทโมดูล (Reset Module)



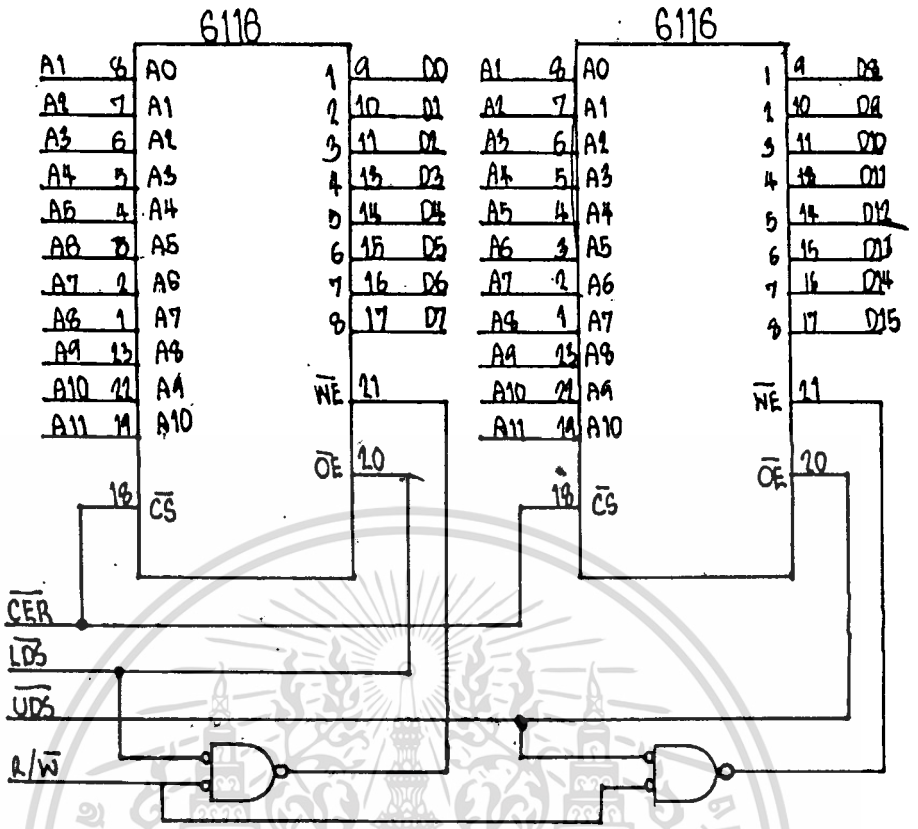
คล็อคโมดูล (Clock Module)



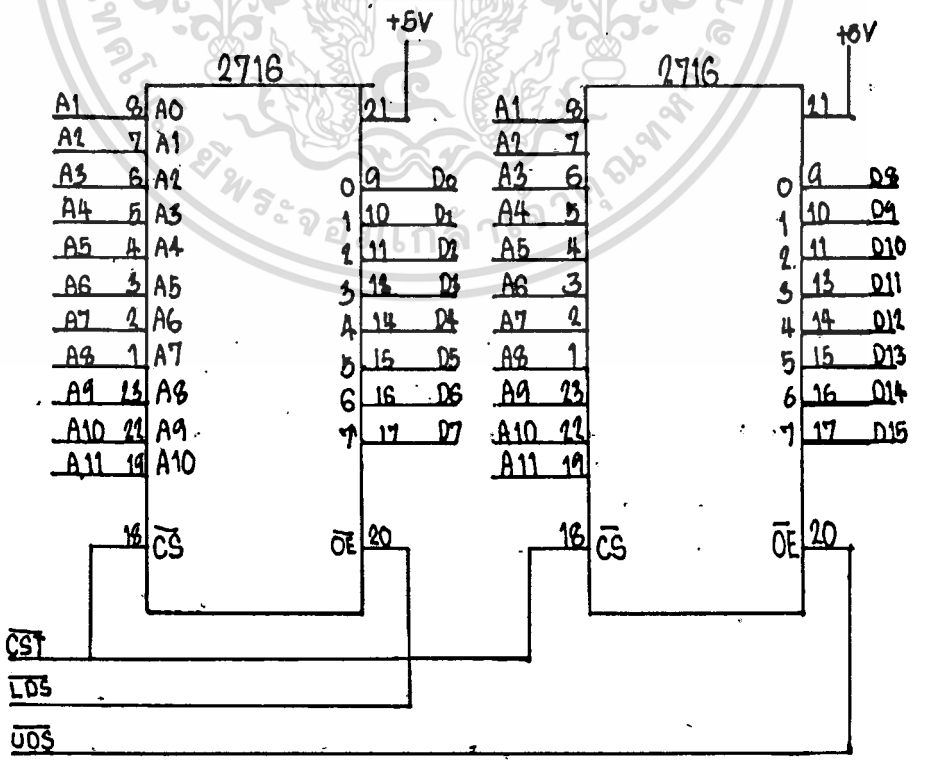
DTACK โมดูล (DTACK Module)



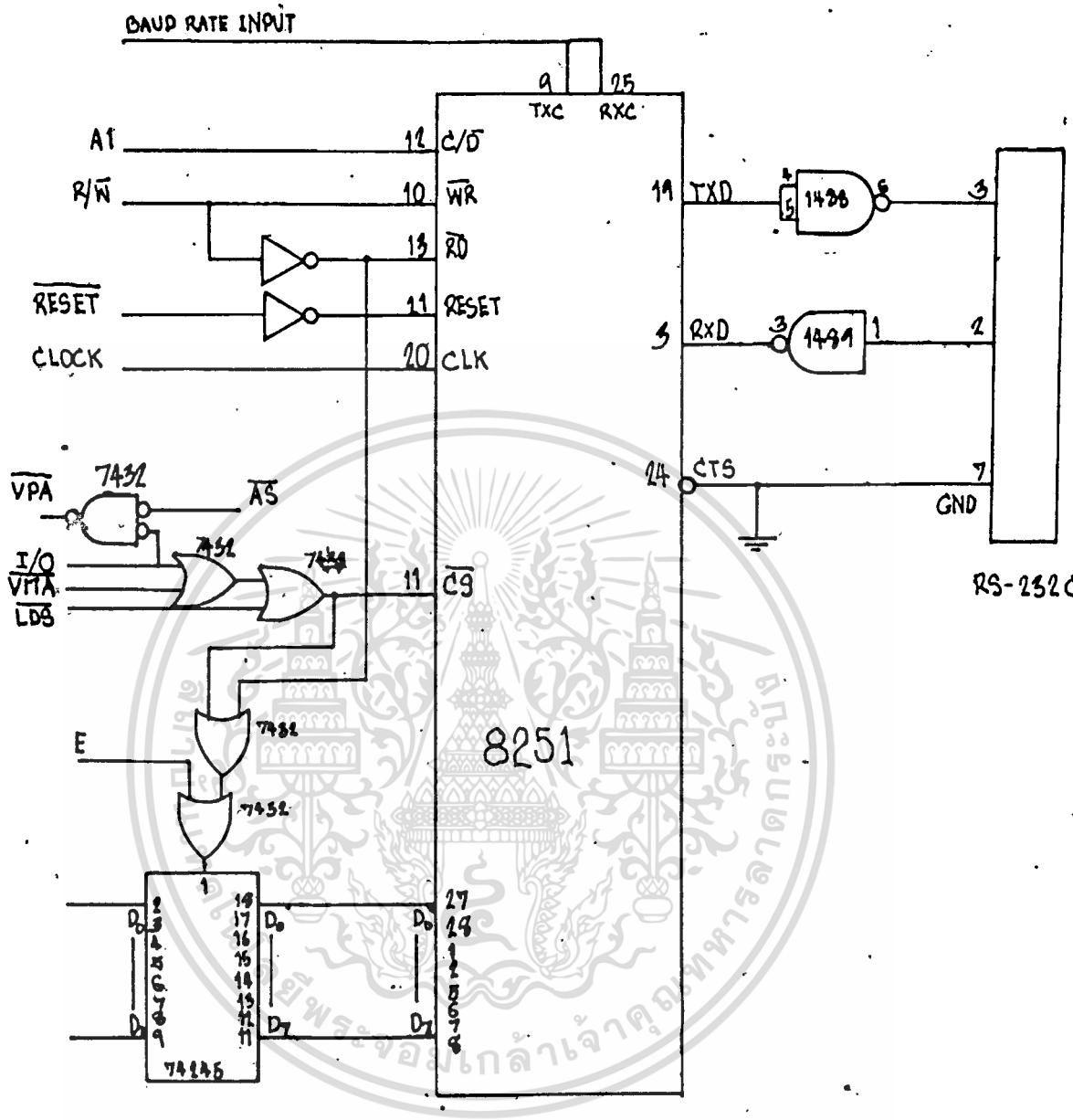
โมดูลถอดรหัส (Decode Module)



แรมโมดูล (RAM Module)



รอมโมดูล (ROM Module)



I/O โมดูล (I/O Module)

บทที่ 4

การพัฒนาทางด้านซอฟท์แวร์ (Software Development)

ในการเขียนโปรแกรมให้ 68000 ทำงานได้นั้นจำเป็นที่จะต้องศึกษาระบบภาษาแอสเซมบลี้ (Assembly Language) ภาษาแอสเซมบลี้ของ 68000 ได้ถูกออกแบบมาเพื่อให้สะดวกและง่ายต่อการเขียนการอ่าน นอกจากนั้นแล้วแต่ละคำสั่งยังมีประสิทธิภาพมากอีกด้วย ซึ่งจุดเด่นต่างๆ ของ 68000 มีดังต่อไปนี้คือ

1. สามารถเขียนให้เลียนแบบภาษาระดับสูง (High Level Language) ได้

การใช้รีจิสเตอร์ของ 68000 นั้นได้แยกเป็นแอดเดรสรีจิสเตอร์ และดาต้ารีจิสเตอร์ ซึ่งเป็นลักษณะการใช้งานเฉพาะที่เป็นข้อดี ซึ่งทำให้เข้าใจโปรแกรม และสามารถเลือกใช้รีจิสเตอร์ได้เด่นชัดมาก สามารถที่จะเลือกใช้ขนาดของข้อมูลได้โดยอิสระ ในแต่ละคำสั่งสามารถแยกออกเป็น 8 บิท, 16 บิท และ 32 บิท

นอกจากนั้นยังมีโหมดในการแอดเดรส หรือการเลือกใช้การอ้างอิงหน่วยความจำ (Memory) ได้ถึง 14 โหมดในแต่ละคำสั่ง ทำให้การใช้คำสั่งเพียงคำสั่งเดียวได้อย่างมีประสิทธิภาพสูงสุด และมีความสามารถใกล้เคียงหรือเทียบเท่ากับภาษาชั้นสูงได้

2. มีคำสั่งที่มีลักษณะคล้ายโมดูล (Modular Structure) อยู่ในตัวเอง

การเขียนโปรแกรมในภาษาแอสเซมบลี้ จะมีโปรแกรมในบางส่วนคล้ายหรือเหมือนกันเป็นส่วนมาก ดังนั้นเวลาเขียนโปรแกรมทำให้สิ้นเปลืองเวลาและเนื้อที่ภายในหน่วยความจำมาก ในระบบของ 68000 จึงได้นำลักษณะของโปรแกรมเช่นนี้มารวบรวมเขียนไว้เป็นคำสั่งเดียว เพื่อให้สะดวกในการใช้งาน เช่น คำสั่ง MOVEM เป็นการเคลื่อนย้ายข้อมูลไปเก็บไว้ในสแตคได้อย่างมีประสิทธิภาพ โดยสามารถเก็บข้อมูลทุกรีจิสเตอร์ลงในสแตค (Push Stack) ได้ทั้งหมดทุกรีจิสเตอร์ภายในคำสั่งเดียว โดยเมื่อเปรียบเทียบกับระบบ

เก่าหรือซีพียูอื่นๆ ต้องใช้ไม่ต่ำกว่า 10 คำสั่ง (Instruction)

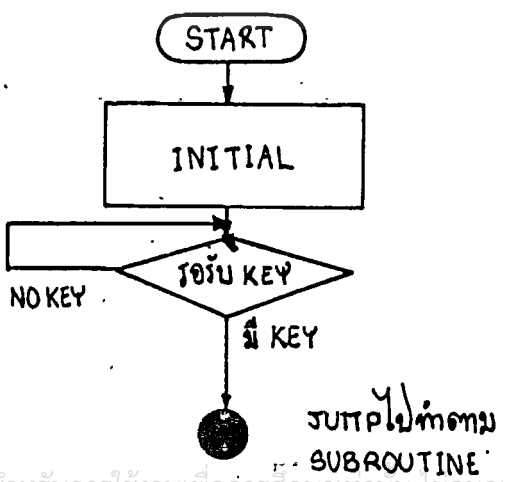
3. การตรวจสอบความผิดพลาด (Error Detection)

ความสามารถในการตรวจสอบความผิดพลาดนี้ เป็นความสามารถเฉพาะของซีพียูสมัยใหม่ เช่น 8088/86 โดยเมื่อเกิดความผิดพลาด จะทำให้เกิดการอินเทอร์รัพท์ แต่ในระบบของ 68000 จะมีความพิเศษ โดยมันจะเข้าไปอยู่ในโหมดของซุปเปอร์ไวเซอร์ ซึ่งจะใช้โหมดนี้เป็นตัวควบคุม มีข้อดีตรงที่จะทำให้ระบบเสียหายมากนัก หรือป้องกันความผิดพลาดที่รุนแรง เช่น ซีพียูเกิดอาการไม่ทำงานหรือตอบรับใดๆ (Hang) การตรวจสอบความผิดพลาดมีอาทิเช่น

- การเข้าหา (Access) ในตำแหน่งที่เป็นแอดเดรสคี่
- ใช้คำสั่งผิด (Illegal Instruction)
- การความผิดพลาดเกี่ยวกับบัส (Bus Error)
- การหารด้วยศูนย์ (Divide by Zero)

4.1 โปรแกรมมอนิเตอร์ (Monitor Program)

โปรแกรมมอนิเตอร์ เป็นโปรแกรมที่ทำหน้าที่เป็นระบบปฏิบัติการ (Operating System) ของบอร์ด 68000 โดยในที่นี้มันจะทำงานในลักษณะรอรับ (Pulling), การกดคีย์ ซึ่งส่งผ่านมาทางอาร์เอส-232 มีโฟลว์ชาร์ตของลักษณะการทำงานอย่างคร่าวๆ ดังรูป 4.1



รูปที่ 4.1

การรับส่งข้อมูลทั้งหมดจะอยู่ในรูปของรหัสที่เป็นตัวหนังสือหรือรหัสแอสกี้ (Ascii code) ซึ่งจะมีโปรแกรมย่อย (Subroutine) ทำหน้าที่เป็นตัวแปลงรหัสแอสกี้นี้ (Ascii) ให้เป็นเลขฐาน 16 เพื่อใช้ในการคำนวณ จุดประสงค์ที่ใช้การรับและส่งเป็นรหัสแอสกี้ ก็เพื่อที่จะต้องการให้สามารถใช้กับดัมพ์เทอร์มินอล (Dump Terminal) ได้ เนื่องจากดัมพ์เทอร์มินอลมีหน้าที่ส่งรหัสที่เป็นตัวอักษรนี้ เมื่อมีการกดคีย์ และทำหน้าที่รับรหัสตัวอักษรเพื่อนำไปแสดงผลบนจอภาพ ส่วนรายละเอียดของโปรแกรมมอนิเตอร์ก็จะแสดงได้ตามโฟลว์ชาร์ต ดังรูปที่ 4.2, 4.3 และ 4.4

การใช้บอร์ด 68000

การใช้บอร์ด 68000 จะต้องมีเทอร์มินอลเครื่องหนึ่ง เพื่อใช้เป็นอินพุทและเอาท์พุทของระบบ หรือใช้โปรแกรมจำลอง (Simulate) ไมโครคอมพิวเตอร์ให้เป็นดัมพ์เทอร์มินอล ซึ่งในที่นี้ใช้เครื่องไอบีเอ็ม พีซี แต่จะต้องระวังในการตั้งอัตราบอร์ด (Board rate) ให้ตรงกัน

ตอนที่จะรันโปรแกรมนี้ ควรจะทำการต่อสายอาร์เอส-232 แบบดีเซลล์(D-Shell) ซึ่งเป็นการส่งผ่านข้อมูลแบบอนุกรมให้กับการ์ด (Card) บนไอบีเอ็ม พีซี จากนั้นทำการรันโปรแกรมจำลองดัมพ์เทอร์มินอล เมื่อเสร็จเรียบร้อยแล้วจึงทำการเปิดแหล่งจ่ายไฟให้บอร์ด 68000 ทำงาน ถ้าระบบสามารถทำงานได้ ก็จะพิมพ์ข้อความดังนี้คือ

"M68000 System Monitor"

และจะเว้นไป 1 บรรทัด แล้วพิมพ์ "-" (Prompt) เพื่อเป็นการรอรับคำสั่งต่อไป ซึ่งมีคำสั่งที่สามารถใช้งานได้ดังนี้

1. "D" ตามด้วย "W" , "B" หรือ "L" เพื่อใช้บอกขนาด แต่ถ้าหากไม่บอก จะถูกเซ็ท (Set) ให้เป็น "W" นอกจากนี้สามารถกำหนดแอดเดรสต้นและแอดเดรสสุดท้ายได้ และถ้าต้องการหยุด ก็กดคีย์ "Ctrl" กับ "C" แต่ถ้าต้องการยกเลิกก็กดคีย์ "Ctrl" กับ "O"

2. "S" ตามด้วย "W" , "B" หรือ "L" เพื่อใช้บอกขนาด หรือถ้าไม่บอกก็จะถูกเซ็ทเป็น "W" และสามารถที่จะกำหนดแอดเดรสได้ ถ้าต้องการหยุดก็ให้กดรหัส "OD" (<CR> : Return)

3. "X" ตามด้วยชื่อของรีจิสเตอร์ เพื่อดูค่าภายในรีจิสเตอร์ที่ต้องการ และ สามารถแก้ไขได้ แต่ถ้ากดเฉพาะ "X" เพียงอย่างเดียว จะแสดงค่าภายในรีจิสเตอร์ทั้งหมดที่มีอยู่ออกมา

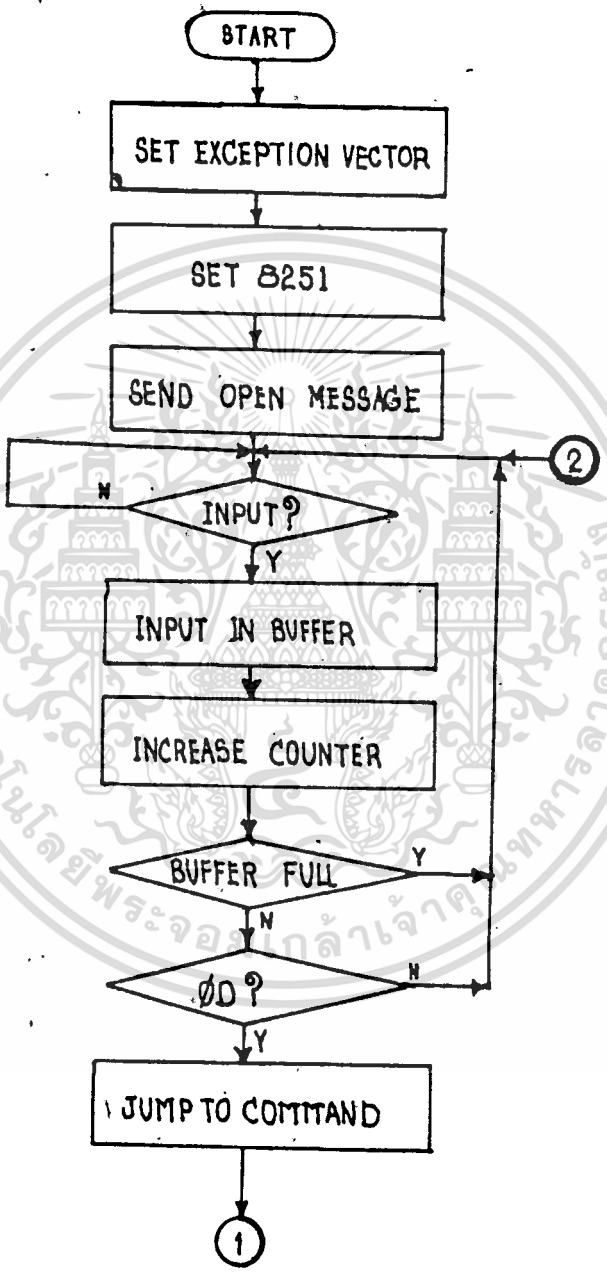
4. "G" เพื่อทำการรัน โดยสามารถกำหนดจุดเริ่มต้นและจุดสุดท้ายได้

5. "T" เพื่อประมวลผลขั้นตอนเดียว (Single Step) ตามด้วยตัวเลขเพื่อบอกว่าต้องการทำกี่ขั้นตอน หรือกี่คำสั่ง ก็จะแสดงผลทุกขั้นตอนที่ได้กำหนด

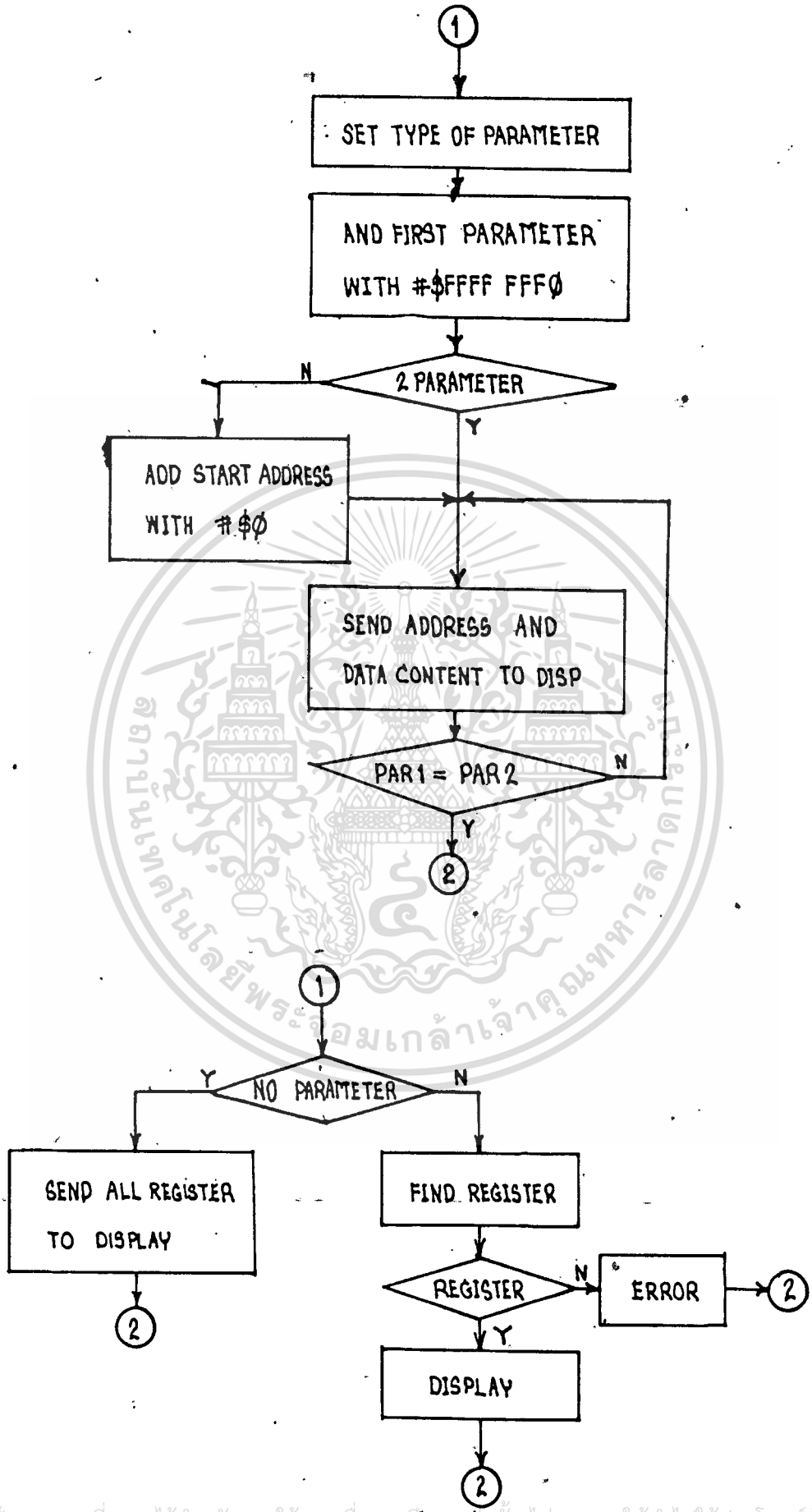
6. "B" จะทำการประมวลผลขั้นตอนเดียว เช่นเดียวกับ "T" แต่จะแสดงผลหลังจากถึงขั้นตอนสุดท้ายแล้ว

7. "A" ตามด้วยค่าแอดเดรส ใช้ในการแปลงคำสั่งของ 68000 แล้วเก็บค่ารหัสแม็ชชีนที่แปลงได้ไว้ในแอดเดรสที่กำหนดนั้น แต่ถ้าไม่ระบุค่าแอดเดรส ก็จะกำหนดให้แอดเดรสเริ่มต้นที่แอดเดรส 300H เมื่อต้องการจะยกเลิกก็ให้กด "."

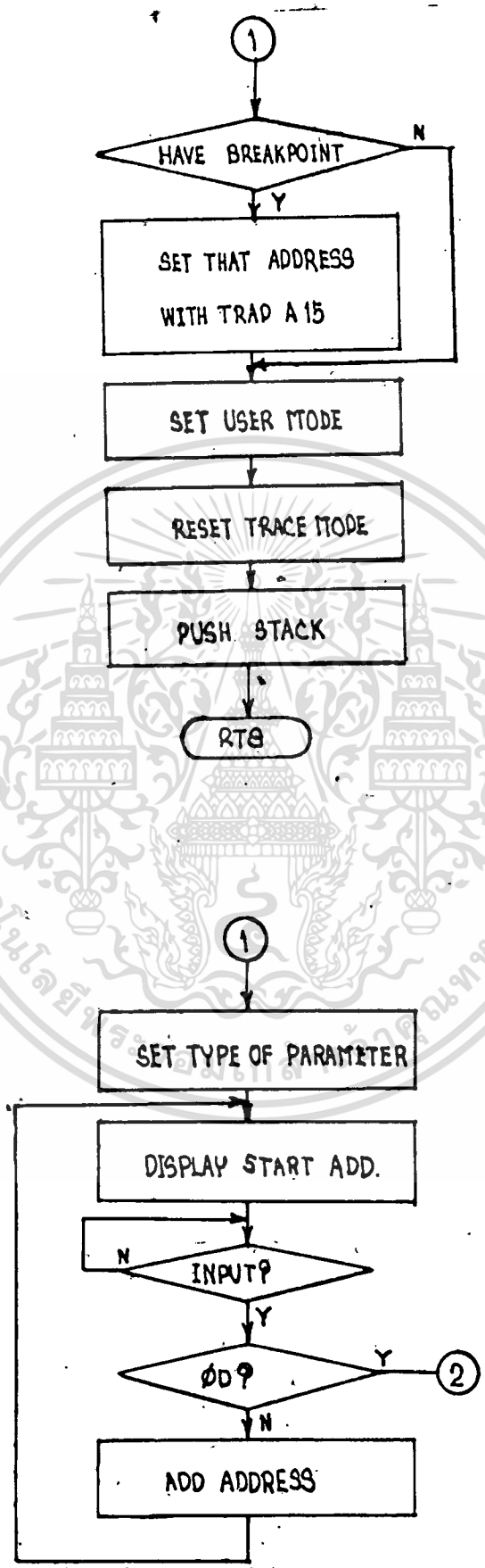
8. "U" ตามด้วยค่าแอดเดรส ใช้ในการแปลงรหัสแม็ชชีนกลับมาเป็นคำสั่งของ 68000 โดยจะทำการแสดงผลทั้งหมดจำนวน 15 คำสั่ง นับจากแอดเดรสที่กำหนด แต่ถ้าไม่ได้ระบุค่าแอดเดรส ก็จะกำหนดให้เริ่มต้นที่แอดเดรส 300H



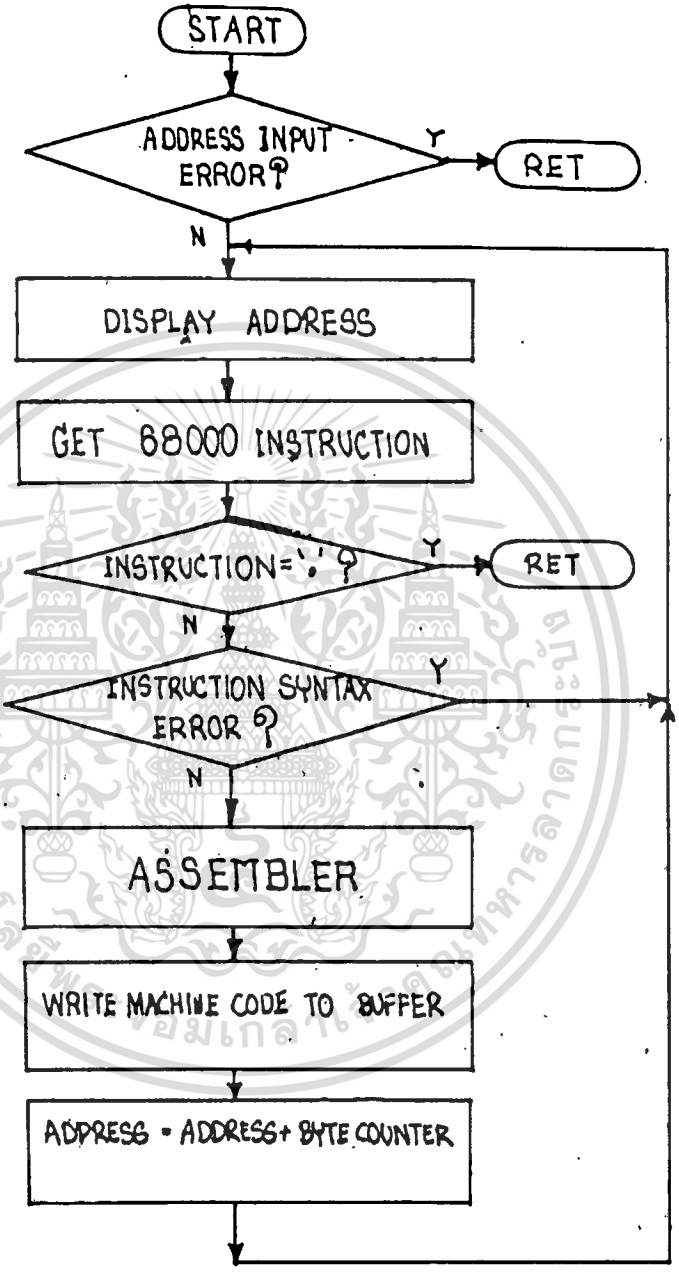
รูปที่ 4.2



รูปที่ 4.3



รูปที่ 4.4



รูปที่ 4.5

4.2 โปรแกรมแอสเซมเบลอร์และดิสแอสเซมเบลอร์ในดีบัก

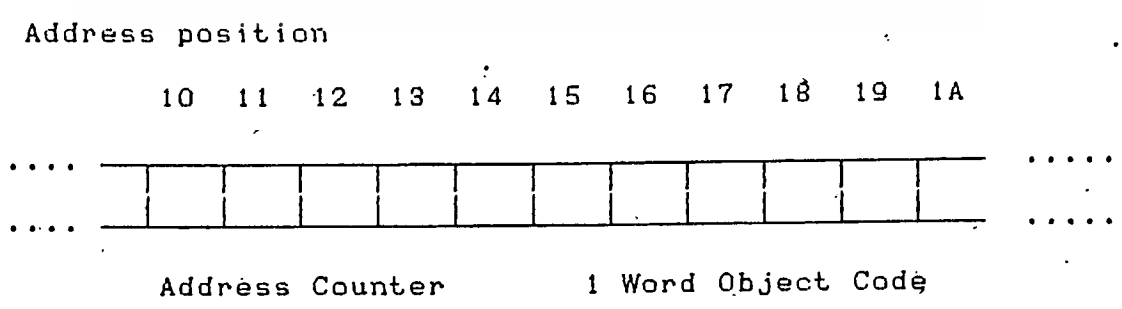
(Assembler and Disassembler Program in Debug)

โปรแกรมแอสเซมเบลอร์และดิสแอสเซมเบลอร์ในดีบักนี้ เป็นโปรแกรมที่สนับสนุนในส่วนของมอนิเตอร์โปรแกรม ซึ่งมีรายละเอียดและขั้นตอนการทำงานต่างๆ ดังนี้ คือ

4.2.1 โปรแกรมแอสเซมเบลอร์ในดีบัก

เป็นโปรแกรมที่ทำหน้าที่แปลคำสั่งของ 68000 ที่ผู้ใช้ป้อนเข้ามาทีละคำสั่งหรือทีละบรรทัด (line) ให้เป็นรหัสแม็ชชีน แล้วเก็บลงในหน่วยความจำสำรอง (Buffer) ซึ่งเป็นลักษณะของการแปลภาษาแบบที่เรียกกันว่า "Interpreter" มีลักษณะการทำงานตามโฟลว์ชาร์ต ดังรูปที่ 4.5

ในขั้นแรกจะมีการตรวจสอบค่าแอดเดรสที่ผู้ใช้ป้อนเข้ามาว่าถูกต้องหรือไม่ เพื่อใช้ในการกำหนดแอดเดรสที่จะใช้เก็บรหัสแม็ชชีนที่แปลได้และจัดเก็บในลักษณะที่ต่อเนื่องกัน โดยมีตัวนับแอดเดรส (Address Counter) ทำหน้าที่เป็นพอยน์เตอร์คอยชี้ตำแหน่งแอดเดรส ซึ่งจะทำการเลื่อนไปที่ตำแหน่งไหนนั้น ขึ้นอยู่กับขนาดหรือจำนวนไบท์ของแต่ละคำสั่งที่แปลได้ (คำสั่งของ 68000 จะมีความยาวหรือขนาดของรหัสแม็ชชีน ตั้งแต่ 2 ไบท์ถึง 10 ไบท์) ดังรูปที่ 4.6



รูปที่ 4.6

เมื่อโปรแกรมทำการอ่านคำสั่งเข้ามาแล้ว ก็จะส่งผ่านมายังโมดูลแอสเซมเบลอร์ โดยที่โมดูลนี้มีรูปแบบการทำงานดังนี้ คือ

จะทำการแยกคำสั่งที่ผู้ใช้ป้อนเข้ามาออกเป็น 3 ส่วนคือ ส่วนนีโมนิค (Mnemonic), ส่วนโอเปอร์แรนด์ตัวต้น (Source Operand) และส่วนโอเปอร์แรนด์ตัวปลาย (Destination Operand) ต่อจากนั้นจึงทำเข้ารหัสเพื่อหาค่าอ๊อบโค้ดในส่วนต่างๆ ที่ทำการแยกแล้วนี้ทีละส่วน แบ่งการทำงานเป็น 2 ชุดคือ

1. ในส่วนนิโมนิค จะถูกนำไปเปรียบเทียบกับตารางค้นหา (Lookup Table) เพื่อหาว่ามีคำสั่งนี้หรือไม่ ถ้าพบก็จะได้อ๊อบโค้ดในส่วนนี้ออกมา ซึ่งเป็นค่าในเลขฐาน 16 และเป็นอ๊อบโค้ดหลัก ให้ชื่อว่า นีโมนิคอ๊อบโค้ด (Mnemonic Opcode) แต่ถ้าพบว่าไม่มีก็จะผ่านไปยังโปรแกรมย่อยที่แสดงผลความผิดพลาด (Error Display)

2. ในส่วนโอเปอร์ตัวต้น และส่วนโอเปอร์แรนด์ตัวปลายจะถูกนำไปตรวจสอบรูปแบบของโอเปอร์แรนด์ว่า อยู่ในโหมดการแอดเดรสเหล่านี้หรือไม่ ดังนี้

- ตรวจสอบว่า อักษรตัวแรกเป็น "#" หรือ "#$" และตัวต่อไปเป็นตัวเลขหรือไม่ (ในภาษาแอสเซมบลี้ของ 68000 "$" จะบอกว่าเป็นตัวเลขฐาน 16) ถ้าใช่ แสดงว่าเป็นโหมดอิมมิเดียทดาต้า ก็ให้เพิ่มค่าตัวนับแอดเดรสไป 2 ค่า และเก็บค่าข้อมูลตัวเลขที่ได้เป็นอ๊อบโค้ดตัวถัดไป แต่ถ้าหากไม่ใช่ ก็ให้ไปตรวจสอบโหมดต่อไป

- ตรวจสอบว่า อักษรตัวแรกเป็น "$" และตัวต่อไปเป็นตัวเลขหรือไม่ ถ้าใช่ แสดงว่าเป็นโหมดแอ็บโซลูท ก็ให้เพิ่มค่าตัวนับแอดเดรสไป 2 ค่า หรือ 4 ค่า โดยขึ้นอยู่กับขนาดของข้อมูลว่าเป็นเวิร์ด หรือลอง แล้วเก็บค่าข้อมูลตัวเลขที่ได้นี้เป็นอ๊อบโค้ดตัวถัดไป แต่หากไม่ใช่ก็ให้ไปตรวจสอบโหมดต่อไป

- ตรวจสอบโดยใช้ตารางค้นหา สำหรับ 5 โหมดต่อไปนี้
  - โหมดดาต้ารีจิสเตอร์ไดเร็คท์
  - โหมดแอดเดรสรีจิสเตอร์ไดเร็คท์
  - โหมดแอดเดรสรีจิสเตอริอินไดเร็คจ์
  - โหมดแอดเดรสรีจิสเตอร์อินไดเร็คจ์ด้วยพรีดีครีเมนท์

- โหมดแอดเดรสรีจิสเตอร์อินไดเร็คจ์ด้วยโพสท์อินคริเมนท์

ถ้าหากตรวจสอบแล้วพบว่าไม่มีในตาราง ก็จะผ่านไปยังโปรแกรมย่อยที่แสดงผลความผิดพลาดที่เกิดขึ้น

ในส่วนของโอเปอร์แรนด์นี้ตรวจสอบแล้วถูกต้อง ก็จะได้อ๊อบโค้ดของโอเปอร์แรนด์ ซึ่งเป็นเลขฐาน 16 ให้ชื่อว่า โอเปอร์แรนด์อ๊อบโค้ด (Operand Opcode)

หลังจากที่ได้ค่าอ๊อบโค้ดทั้ง 3 ส่วนแล้ว ก็จะนำค่าทั้ง 3 คือ นิโมนิคอ๊อบโค้ด โอเปอร์แรนด์อ๊อบโค้ดของโอเปอร์แรนด์ตัวต้น และของโอเปอร์แรนด์ตัวปลาย มารวมเข้าด้วยกัน (Merge) ก็จะได้รหัสแม็ชชีนซึ่งเป็นอ๊อบโค้ดหลักของคำสั่งนั้น

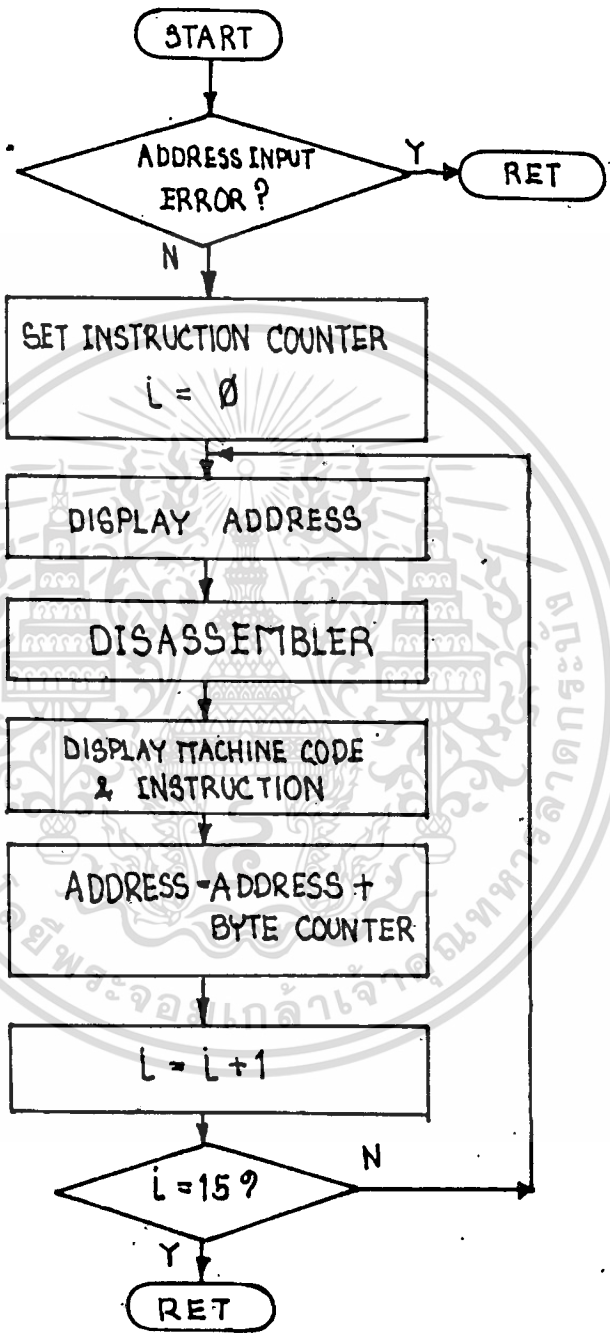
4.2.2 โปรแกรมดิสแอสเซมเบลอร์ในดีบัก

เป็นโปรแกรมที่ทำหน้าที่ตรงข้ามกับโปรแกรมแอสเซมเบลอร์ โดยจะทำการแปลรหัสแม็ชชีนที่อยู่ในแอดเดรสที่ผู้ใช้กำหนด กลับมาเป็นคำสั่งของ 68000 มีลักษณะการทำงานตามโฟลว์ชาร์ตดังรูปที่ 4.7

โดยขั้นตอนแรกจะทำการตรวจสอบแอดเดรสที่ผู้ใช้ป้อนเข้ามาก่อนว่า ถูกต้องหรือไม่ ถ้าพบว่าถูกต้อง จึงทำการแสดงผลแอดเดรสนั้นออกจอภาพ ต่อจากนั้นก็ผ่านไปยังโมดูลดิสแอสเซมเบลอร์โดยมีรูปแบบการทำงานดังต่อไปนี้ คือ

จะทำการ AND/OR บิทระหว่างรหัสแม็ชชีนตามแอดเดรสที่กำหนด กับข้อมูลที่เป็นส่วนนิโมนิคของคำสั่งในตารางค้นหา ถ้าพบ ก็จะทำการตรวจสอบโหมดในการแอดเดรส ว่าอยู่ในโหมดไหน โดยวิธีการ AND/OR บิท เช่นเดียวกัน

- ถ้าหากเป็นโหมดดังต่อไปนี้ ก็จะได้คำสั่งที่แท้จริง ได้แก่
  - โหมดดาต้ารีจิสเตอร์ไดเร็คท์
  - โหมดแอดเดรสรีจิสเตอร์ไดเร็คท์
  - โหมดแอดเดรสรีจิสเตอริอินไดเร็คจ์
  - โหมดแอดเดรสรีจิสเตอร์อินไดเร็คจ์ด้วยพรีดีครีเมนท์



รูปที่ 4.7

- โหมดแอดเดรสรีจิสเตอร์อินไดเรคต์ด้วยโพสท์อินคริเมนท์

- ถ้าหากอยู่ในโหมดดังนี้ จะต้องอ่านค่าอ๊อบโค้ดในเวิร์ดถัดไป ด้วย เพื่อให้ได้คำสั่งที่สมบูรณ์ ได้แก่

- โหมดอิมมิเดียทดาต้า

- โหมดแอ็บโซลูท

4.3 โปรแกรมคอมไพเล่อร์ของ 68000 แอสเซมเบลอร์

แบ่งขั้นตอนออกเป็น 3 พาส (Pass) ด้วยกัน คือ

- พาสที่ 1

จุดประสงค์ของพาสแรกนี้ จะนำ 68000 รหัสต้นแบบ (68000 Source Code) มาสร้างเป็นตารางสัญลักษณ์ (Symbol Table) โดยแต่ละคำสั่งจะถูกสแกน (Scan) ทีละคำสั่ง และมีตัวนับแอดเดรส (Address Counter) ซึ่งจะขึ้นอยู่กับความยาวของคำสั่งนั้นๆ (ความยาวตั้งแต่ 2 ไบท์ถึง 10 ไบท์) และเมื่อพบลาเบล (Label) ค่าของตัวนับแอดเดรสขณะนั้นและชื่อลาเบลนั้นก็จะถูกเก็บไว้ในตาราง Label_tab

ถ้าพบคำสั่งประเภท Jump หรือ Branch เช่น JMP , BGE เป็นต้น โดยจะทำการเก็บคำสั่งนั้นทั้งบรรทัดลงในตาราง Jmp_tab รวมทั้งแอดเดรสของคำสั่งนั้น, ตำแหน่งของบรรทัดนั้นและจะต้องเก็บค่าอ๊อบโค้ด, ความยาว และตำแหน่งแอดเดรสที่สมมติขึ้นลงในตาราง Opc_tab เสียก่อน เพื่อความสะดวกในการกำหนดตำแหน่งของคำสั่งในภายหลัง

ส่วนคำสั่งที่เหลือทั้งหมด ก็จะทำการแปลให้เป็นอ๊อบโค้ด, ความยาว, ตำแหน่งแอดเดรสของคำสั่งลงในตาราง Opc_tab

- พาสที่ 2

จะทำการเปลี่ยนชื่อของลาเบลให้เป็นแอดเดรสโดยใช้ตาราง Label_tab หลังจากนั้นก็จะมาทำงานที่ตาราง Jmp_tab โดยการเปลี่ยนลาเบลที่จะ Jump เป็นแอดเดรสแทน เพื่อความสะดวกในการแปล (Compile) จากนีโมนิคมาเป็นอ๊อบโค้ด เพราะฉะนั้นเมื่อผ่านการแปลแล้วก็จะได้อ๊อบโค้ดและความยาวที่แท้จริงลงไปเก็บในตาราง Opc_tab แทนค่าที่สมมติขึ้น พร้อม

เซ็ทแฟล็ก (Flag) เพื่อบ่งบอกถึงการเพิ่มขึ้นของแอดเดรสที่แท้จริง หลังจากที่ได้ผ่านการสมมติมา แต่ยังไม่ได้เปลี่ยนแปลงแอดเดรสเหล่านั้นจริง เพียงแต่เซ็ทแฟล็กไว้เท่านั้น เพื่อรอในพาสที่ 3 ต่อไป

- พาสที่ 3

จะเห็นได้ว่าแอดเดรสที่มีอยู่ในตาราง Opc_tab ในขณะนี้จะเป็นแอดเดรส ที่ไม่แท้จริง เพราะว่าในพาสที่ 1 เราสมมติให้บรรทัดที่มีคำสั่งประเภท Jump กับ Branch มีความยาวของอ๊อบโค้ดเท่ากับ 2 ซึ่งเป็นความยาวของอ๊อบโค้ดที่น้อยที่สุด แล้วบวกแอดเดรสต่อไปด้วยความยาวเท่ากับ 2 ซึ่งถ้าความยาวของอ๊อบโค้ดนั้นๆ มากกว่า 2 แล้ว แอดเดรสที่กำหนดไว้ก็จะผิด เพราะฉะนั้นจะต้องบวกค่าแอดเดรสเหล่านั้นด้วยแฟล็กที่ถูกเซ็ทไว้ในพาสที่ 2 ก็จะเป็นแอดเดรสที่ถูกต้องโดยสมบูรณ์ แล้วเก็บไว้ในตาราง Opc_tab

บทที่ 5

บทวิจารณ์และสรุป

โครงงานนี้เกิดจากความสนใจในคุณสมบัติพิเศษของไมโครโปรเซสเซอร์ของ 68000 แบบ 16 บิทที่มีโครงสร้างภายในเป็นแบบ 32 บิท สามารถรับส่งข้อมูลได้ทั้งแบบซิงโครนัสและอะซิงโครนัส (Synchronous and Asynchronous) ทำให้มีความเร็วในการทำงานสูง และสามารถปรับให้เข้ากับอุปกรณ์ (Device) ซึ่งมันจะติดต่อได้ด้วย นอกจากนั้นยังมีคำสั่งที่ประสิทธิภาพสูง (Powerfull) และวิธีการอ้างถึงหน่วยความจำ ให้เลือกใช้ได้อย่างสะดวก

การศึกษาคุณสมบัติของซีพียู (CPU) จำเป็นต้องทำในรูปของวงจรแผ่นพิมพ์เดียว (Single Board) ซึ่งทำให้ศึกษาได้ทั้งฮาร์ดแวร์และซอฟท์แวร์ (Hardware and Software) โดยจะมีลักษณะพิเศษแตกต่างจากวงจรแผ่นพิมพ์เดียวของไมโครโปรเซสเซอร์อื่นทั่วๆ ไป เพราะจะมีเทอร์มินอลทำหน้าที่เป็นอุปกรณ์อินพุทและเอาท์พุท ซึ่งส่งผ่านข้อมูลแบบอนุกรมทางพอร์ตมาตรฐานอาร์เอส-232

โครงงานนี้สามารถแบ่งส่วนต่างๆ ออกได้ดังนี้

1. ส่วนวงจรหลัก (Main Board) จะประกอบด้วยซีพียู 68000 หน่วยความจำ พอร์ตอนุกรม (Serial Port)

2. ส่วนโปรแกรมควบคุมการทำงานของวงจรหลัก มีหน้าที่ทำการประมวลผล และควบคุมวงจรหลัก

3. ส่วนโปรแกรมดีบักเกอร์ซึ่งใช้ความสามารถของไอบีเอ็ม พีซีในการรับส่งข้อมูลไปให้ 68000 บนวงจรหลัก โดยมีโปรแกรมแอสเซมเบลอร์ทำหน้าที่แปลคำสั่ง 68000 ให้เป็นรหัสแม็ชชีน และโปรแกรมดิสแอสเซมเบลอร์ทำหน้าที่แปลรหัสแม็ชชีนกลับมาเป็นคำสั่ง 68000

4. ส่วนโปรแกรม 68000 คอมไพเล่อร์ ทำหน้าที่เป็นครอสแอสเซมเบลอร์ (Cross Assembler) บนไอบีเอ็ม พีซี

ปัญหาของการทำโครงงานนี้ คือ

- ไม่มีผู้เชี่ยวชาญทางด้านนี้โดยเฉพาะ ดังนั้นจึงต้องทำการศึกษา

และค้นคว้ารายละเอียด ตั้งแต่ขั้นพื้นฐาน ทำให้เสียเวลาในการทดลอง และวินิจฉัยหาข้อผิดพลาด

- การพัฒนาทางด้านซอฟท์แวร์ เป็นไปด้วยความยากลำบาก เนื่องจากไม่มีอุปกรณ์สนับสนุนอย่างเช่นโปรแกรมแอสเซมเบลอร์ของ 68000 และเครื่องไมโครคอมพิวเตอร์ที่มี 68000 เป็นซีพียู

- เนื่องจากการโครงงานนี้จำเป็นต้องใช้อุปกรณ์เฉพาะ ซึ่งมีราคาสูง ทำให้การนำมาใช้งาน จะต้องลงทุนสูงตามไปด้วย ดังนั้นเพื่อจะเป็นการประหยัดต้นทุน จึงจำเป็นต้องหาอุปกรณ์ที่มีราคาไม่สูงนัก และสามารถหาซื้อได้มาทดแทน โดยที่ให้ผลใกล้เคียงกัน เช่นในการใช้ไอซี 8251 แทนไอซี 6850 ซึ่งเป็นการยากที่จะใช้อุปกรณ์ต่างระบบมาใช้ร่วมกัน และจะกินเวลามาก เพราะว่าไม่มีข้อมูลที่เพียงพอ

อย่างไรก็ตาม โครงงานชิ้นนี้เป็นเพียงส่วนหนึ่งซึ่งจะสามารถใช้เป็นแนวทางในการพัฒนาระบบให้สมบูรณ์ยิ่งขึ้นเพราะจะช่วยแก้ไขปัญหาต่างๆ ข้างต้นได้ แต่การใช้งานจริงอาจจะไม่สะดวกนัก ดังนั้นโครงงานนี้จึงต้องการการพัฒนาตามรูปแบบและจุดประสงค์ของผู้ต้องการใช้งาน

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ได้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความร่วมมือจากหลายๆ ฝ่ายเป็นอย่างดี ผู้จัดทำขอขอบคุณ

ดร. บุญวัฒน์ อัตชู

รศ. มนัส สังวรศิลป์

อ. วิริยะ ฉัตรวิริยะ

อ. ดนัยวัฒน์ กิตตินันต์

คุณกิตติวุฒิ เอื้ยบเจริญ

คุณสมประสงค์ กาญจนวัฒนา

คุณวิษณุ ฤดีนรเวท

ที่ได้ให้คำแนะนำ ข้อคิดเห็นบางประการอันก่อประโยชน์แก่การทำวิทยานิพนธ์ฉบับนี้

หนังสืออ้างอิง

ก. เอกสารอ้างอิงที่เป็นวารสารภาษาอังกฤษ จัดเรียงลำดับดังนี้

1. Richard Rodman, " Series 32000 Cross Assembler ", Dr. dobb's Journal, on Articles, December 1986, pp.48-49

2. Richard Rodman, Listing " 32000 Cross Assembler ", Dr. dobb's Journal, January 1987, pp.82-98

ข. เอกสารอ้างอิงที่เป็นวารสารภาษาไทย จัดเรียงลำดับดังนี้

1. ประพันธ์ ธีระวรรณวิไล, "MC 68000 ไมโครโปรเซสเซอร์ 16/32 บิท", วารสารไมโครอิเลคทรอนิคส์, ฉบับที่ 9, หน้า 58-69.

ค. เอกสารอ้างอิงที่เป็นหนังสือภาษาไทยและภาษาอังกฤษ จัดเรียงลำดับดังนี้

1. Steve Schustack, "Variations in C", Microsoft Press, 344 p., 1985.

2. Microsoft Corporation, "Microsoft C Compiler version 4.0", 444 p.,

3. Brian W. Kernighan, Dennis M. Ritchie, "The C Programming Language", Prentice-Hall, INC., 228 p.,

4. Lance A. Leventhal, Doug Hawkins, Gerry Kane, and William D. Cramer, "68000 Assembly Language Programming", McGraw_Hill, 484 p., 1986.

5. Motorola Inc., "MC 68000 16-Bit Microprocessor User's Manual", Prentice, Inc., 1982,1980,1979, 231 p.,