ปีการศึกษา 2531

การพัฒนาระบบไมโครโปรเซสเซอร์เบอร์ 68000

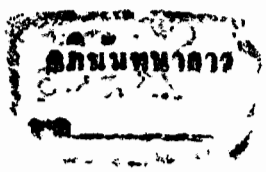
โดย

นางสาวมยุรี ว่องไววิศาล รหัส 281175

นางสาวสุดา ศรีสืบ รหัส 281272

อาจารย์ที่ปรึกษา

ดร.แดเนียล บรีน



ปริญญานิพนธ์ปีการศึกษา 2531

ภาควิชา อิเล็คทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบไมโครโปรเซสเซอร์เบอร์ 68000

ผู้จัดทำ

1. นางสาวมยุรี ว่องไววิศาล 281175 ภาควิชาอิเล็คทรอนิคส์
2. นางสาวสุดา ศรีสืบ 281272 ภาควิชาอิเล็คทรอนิคส์

.....อาจารย์ที่ปรึกษา

(ดร.แดเนียล บรีน)

การพัฒนาระบบของไมโครโปรเซสเซอร์เบอร์ 68000

มยุรี ว่องไววิศาล
สุดา ศรีสืบ
ดร.แดเนียล บรีน อาจารย์ที่ปรึกษา
ปีการศึกษา 2531

บทคัดย่อ

โครงงานนี้แบ่งเป็น 2 ส่วนคือ

1. ฮาร์ดแวร์ โดยการสร้างการ์ดขึ้นมาเสียบที่สล็อตของเครื่องไอบีเอ็ม และให้มีการติดต่อกับผู้ใช้โดยเครื่องไอบีเอ็มจะเป็นส่วนจัดการ การรับส่งข้อมูลผ่านทางคีย์บอร์ดและจอ(คีย์บอร์ดทำหน้าที่เป็นส่วนอินพุท และจอทำหน้าที่เป็นส่วนเอาท์พุท) การ์ดที่สร้างขึ้นจะมีหน่วยความจำสแตติคแรม ขนาด 32 กิโลไบท์ 8 บิท จำนวน 2 ตัว เพื่อที่จะใช้เป็นหน่วยความจำของ 68000 ซึ่งเป็นไมโครโปรเซสเซอร์ 16 บิท และยังต้องมีส่วนมัลติเพลกซ์ ส่วนควบคุมการทำ ดีเอ็มเอ (DMAC) ส่วนอินเตอร์เฟส ส่วนควบคุมบัสและบัฟเฟอร์ โดยการนำวงจรแต่ละส่วนที่ได้ออกแบบมาต่อร่วมกันเป็นระบบ

2. ซอฟท์แวร์ โดยต้องเขียนซอฟท์แวร์ขึ้นเพื่อสนับสนุนการใช้งานของการ์ด ซึ่งมีโปรแกรมแอสเซมเบลอร์ โปรแกรมดีบักเกอร์ และ โปรแกรมโหลดเดอร์ โปรแกรมแอสเซมเบลอร์จะใช้แปลภาษาแอสเซมบลี ของ 68000 เป็นภาษาเครื่อง โปรแกรมดีบักเกอร์จะช่วยในการเขียนโปรแกรมแอสเซมเบลอร์ให้ง่ายและสะดวกขึ้น และ โปรแกรมโหลดเดอร์จะใช้สำหรับโหลดข้อมูลในหน่วยความจำของเครื่องไอบีเอ็ม ไปไว้ในหน่วยความจำของ 68000 บนการ์ด

โครงงานนี้จะเป็นจุดเริ่มต้นและเป็นแนวทาง ในการพัฒนาการนำเอาไมโครโปรเซสเซอร์เบอร์ 68000 มาใช้งานเป็นซีพียู เช่น ในงานควบคุมระบบต่างๆ โดยอาจจะพัฒนาฮาร์ดแวร์ เมื่อนำไปใช้งานจริงๆ ได้โดยต่อวงจรเพิ่มเติมลงบนการ์ดที่ทำในโครงงานนี้ และทดสอบการทำงานของวงจรผ่านเครื่องคอมพิวเตอร์ไอบีเอ็ม โดยใช้ซอฟท์แวร์ต่างๆที่เขียนขึ้นช่วย

68000 Development System

Mayuree Vongvaivisal

Suda Srisoeb

Dr.Daniel Breen Advisor

1988

Abstract

This project is divided into 2 part.

1. Hardware - By building an interface card to be put in the slot of IBM PC computers or compatibles. The user can use keyboard and CRT of the IBM PC as input and output. This card has 2 chips of 32 kilobyte , 8 bit static RAM which is used as memory for the 16 bit microprocessor. Other circuits in this card are multiplexer , DMA controler , interfacer , bus controler and buffer.

2. Software - By writing programs to support the operation of the card.These programs are assembler , debugger and loader. The assembler program is used to translate the assembly language of MC 68000 to machine language. To write the assembler quickly and easily , the debugger is used to help in correcting assembler program. The loader program is used for loading data from memory of the IBM PC to the memory on the card.

This project demonstrates the way in developing the system that used 68000 as CPU. Some examples are control system. That the user can add the necessary hardware on the interface card. The user can test the circuit of this card by using the softwares.

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

ในปัจจุบันเครื่องไมโครคอมพิวเตอร์ ก้าวเข้ามามีบทบาทในสังคมอย่างมากมาย เครื่องไมโครคอมพิวเตอร์ใหม่ๆที่ออกมาก็มีประสิทธิภาพสูงขึ้นตามลำดับ เพราะเนื่องจากความก้าวหน้าทางเทคโนโลยี ทำให้อุปกรณ์ต่างๆมีขีดความสามารถสูงขึ้น

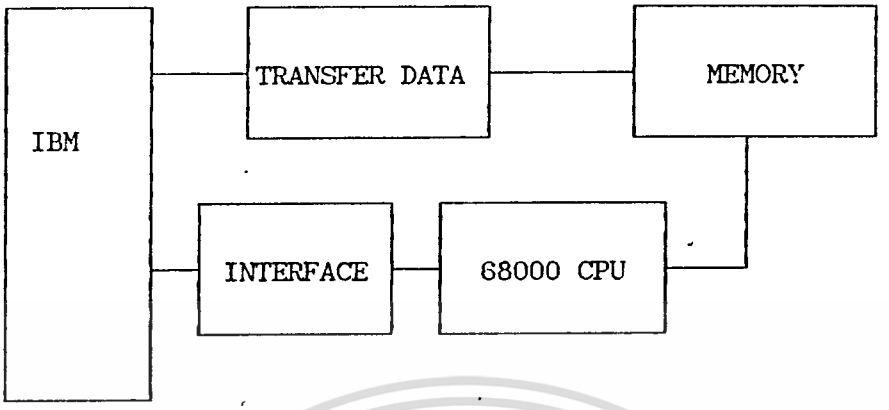
อุปกรณ์ชิ้นหนึ่งซึ่งเป็นหัวใจสำคัญ และมีบทบาทมากที่สุดในเครื่องไมโครคอมพิวเตอร์ ก็คือ หน่วยประมวลผลกลาง (CPU) หรือ ไมโครโปรเซสเซอร์ (Microprocessor)

ไมโครโปรเซสเซอร์เบอร์หนึ่งที่กำลังได้รับความนิยมก็คือ ไมโครโปรเซสเซอร์เบอร์ 68000 ซึ่งเป็นไมโครโปรเซสเซอร์ที่มีประสิทธิภาพในการทำงานสูง เราจึงได้จัดทำโครงงานนี้ขึ้นมา โดยการนำไมโครโปรเซสเซอร์เบอร์ 68000 มาใช้เป็นซีพียู และเรายังต้องเขียนซอฟท์แวร์(Software)ขึ้นมาช่วยในการใช้งานของไมโครโปรเซสเซอร์เบอร์นี้ ดังนั้นโครงงานนี้ถือได้ว่าเป็นโครงงานที่สร้างขึ้นมา เพื่อเป็นจุดเริ่มต้นในการพัฒนาและนำไมโครโปรเซสเซอร์เบอร์ 68000 มาใช้งาน

1.2 ลักษณะของโครงงาน

ลักษณะของโครงงานที่ทำแบ่งเป็น 2 ส่วนใหญ่ๆ คือ ด้านฮาร์ดแวร์ (Hardware) และด้านซอฟท์แวร์ (Software)

ด้านฮาร์ดแวร์เราจะสร้างวงจรบนการ์ด (Card) ที่สามารถติดต่อ และควบคุมการทำงานโดยใช้เครื่องคอมพิวเตอร์ไอบีเอ็ม พีซี/เอ็กซ์ที (IBM PC/XT) โดยโครงสร้างของวงจรที่จัดทำขึ้นนี้ สามารถแบ่งออกเป็นส่วนใหญ่ ๆ คือ ส่วนของเครื่องไอบีเอ็ม พีซี ซึ่งจะทำหน้าที่เป็นเหมือนส่วนที่ทำหน้าที่ติดต่อระหว่างผู้ใช้กับวงจรที่จัดทำขึ้น และส่วนของวงจรก็สามารถแบ่งออกเป็นส่วนต่าง ๆ ก็คือ ส่วนของ ตัวไมโครโปรเซสเซอร์ 68000, ส่วนของหน่วยความจำ, ส่วนที่ทำหน้าที่ติดต่อระหว่างตัวไอบีเอ็ม พีซี กับวงจรบนการ์ด และส่วนที่จัดการการแลกเปลี่ยนข้อมูลระหว่างหน่วยความจำบนการ์ด กับหน่วยความจำที่อยู่ในเครื่องไอบีเอ็ม พีซี ซึ่งส่วนฮาร์ดแวร์นี้สามารถเขียนเป็น บล็อกไดอะแกรม (Block Diagram) ได้ดังรูปที่ 1.1



รูป 1.1 บล็อกไดอะแกรมแสดงส่วนประกอบของฮาร์ดแวร์

ในการติดต่อกับผู้ใช้ จะติดต่อโดยให้เครื่องไมโครคอมพิวเตอร์ทำหน้าที่เป็นส่วนจัดการการรับส่งข้อมูล (I/O) แล้วส่งผ่านต่อให้ซีพียู (CPU) บนการ์ด

สำหรับด้านซอฟแวร์ก็จะแบ่งเป็น 3 ส่วนด้วยกันคือ

1. ส่วนแอสเซมเบลอร์ (Assembler) ซึ่งจะเป็นส่วนที่ทำหน้าที่ในการแปลคำสั่งของไมโครโปรเซสเซอร์ 68000 ให้เป็นรหัสเครื่อง (Machine Code)

2. ส่วนดีบักเกอร์ (Debugger) มีหน้าที่ในการช่วยเขียนโปรแกรม เพราะสามารถตรวจสอบแก้ไขข้อผิดพลาดจากการเขียนโปรแกรมแอสเซมบลีได้ง่าย โดยสามารถตรวจสอบการทำงานทีละคำสั่งได้

3. ส่วนโหลดเดอร์ (Loader) มีหน้าที่นำข้อมูลจากหน่วยความจำของเครื่องไอบีเอ็ม หรือข้อมูลในแผ่นดิสต์ (Diskette) ไปบรรจุลงในหน่วยความจำบนแผ่นการ์ด

สำหรับรายละเอียดต่างๆจะได้กล่าวต่อไป

1.3 แผนการดำเนินงาน

ในโครงงานนี้เราได้แบ่งการดำเนินงานออกเป็น 2 ส่วนคือ

1.3.1 ด้านฮาร์ดแวร์

เราได้ทำการศึกษาระบบโดยเริ่มต้นจากการทดลองวงจรในขั้นพื้นฐานก่อน ในลักษณะของฟรีรันโหมด (Free Run Mode) เพื่อตรวจสอบดูว่าซีพียูที่ใช้สามารถใช้งานได้จริง รวมทั้งมีวงจรสร้างสัญญาณนาฬิกา (Clock Generator) และวงจรรีเซ็ท (Reset) ประกอบด้วย โดยทำการทดสอบวงจรย่อยต่างๆที่เกี่ยวข้องดังกล่าวทีละส่วน ก่อนที่จะนำมาประกอบหรือต่อเข้า

ด้วยกันเป็นระบบ

หลังจากนั้นจึงทำการเพิ่มวงจรย่อยอื่นๆ เช่นวงจรที่เกี่ยวข้องกับหน่วยความจำของระบบ และวงจรย่อยอื่น ๆ อีกบางส่วน เมื่อทดสอบวงจรย่อยเหล่านี้แล้วก็สามารถรวมกันเข้าเป็นระบบต่อไป

1.3.2 ด้านซอฟท์แวร์

1.3.2.1 โปรแกรมแอสเซมเบลอร์ (Assembler Program) จะแปลคำสั่งของ 68000 เป็นรหัสเครื่อง โดยอ่านจากแฟ้มข้อมูล (File) ที่เก็บไว้ในแผ่นดิสต์เพื่อสร้างแฟ้มข้อมูลใหม่ที่เป็นรหัสเครื่องของ 68000

1.3.2.2 โปรแกรมดีบักเกอร์ (Debugger Program) เราจะใช้ดีบักในการตรวจสอบโปรแกรมทีละคำสั่ง เมื่อเราพบข้อผิดพลาดก็สามารถแก้ไขให้ถูกต้องได้

1.3.2.3 โปรแกรมโหลดเดอร์ (Loader Program) เป็นโปรแกรมที่เขียนขึ้นมาเพื่อใช้ในการนำข้อมูลในแผ่นดิสค์ หรือ ในหน่วยความจำของเครื่องไอบีเอ็มไปโหลดไว้ในหน่วยความจำที่อยู่บนการ์ด โดยโปรแกรมที่เขียนขึ้นมาบนเครื่องไอบีเอ็มสามารถถูกเรียกใช้เพื่อโหลดโปรแกรมของ 68000 ลงบนการ์ดได้โดยง่าย

1.4 ขอบเขตโครงงาน

- ออกแบบและสร้างวงจรที่ใช้ไมโครโปรเซสเซอร์เบอร์ 68000 เป็นซีพียู โดยใช้เครื่องไอบีเอ็มเป็นตัวจัดการอินพุท/เอาท์พุท (Input/Output)

- เขียนโปรแกรมแอสเซมเบลอร์ของ 68000 เพื่อใช้ในการแปลงภาษาแอสเซมบลีเป็นภาษาเครื่อง

- เขียนโปรแกรมดีบักเกอร์เพื่อช่วยในการเขียนโปรแกรมแอสเซสเบลอร์ให้ง่ายและสะดวกขึ้น

- เขียนโปรแกรมโหลดเดอร์ ทำหน้าที่จัดการการแลกเปลี่ยนข้อมูลระหว่างหน่วยความจำบนการ์ดกับหน่วยความจำบนเครื่องไอบีเอ็ม พีซี

1.5 ประโยชน์ที่คาดว่าจะได้รับ

โครงงานนี้ สามารถนำไปใช้เป็นเครื่องมือในการพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์เบอร์ 68000 เป็นซีพียู โดยถือว่าเป็นจุดเริ่มต้นในการพัฒนา และเป็นแนวทางสำหรับผู้สนใจที่จะนำความรู้และแนวความคิดต่างๆที่ได้จากโครงงานนี้ ไปสร้างการ์ดที่มี 68000 เป็นซีพียูขึ้นมา และในโครงงานนี้เราจะได้เขียนซอฟท์แวร์ขึ้นมาด้วย เพื่อเป็นการสนับสนุนการใช้งานของการ์ด

ดังนั้นผู้ที่สนใจสามารถนำไปเป็นแนวทางได้ โดยผู้ที่มีเครื่องไอบีเอ็มอยู่แล้ว สามารถนำการ์ดที่สร้างขึ้นมาไปใช้งานได้เลย โดยที่แป้นพิมพ์ (Keyboard) จะเป็นส่วนอินพุทของข้อมูล และจอภาพจะเป็นส่วนเอาท์พุท ซึ่งการทำงานบนการ์ดจะสามารถทำได้รวดเร็วและมีประสิทธิภาพสูง

บทที่ 2

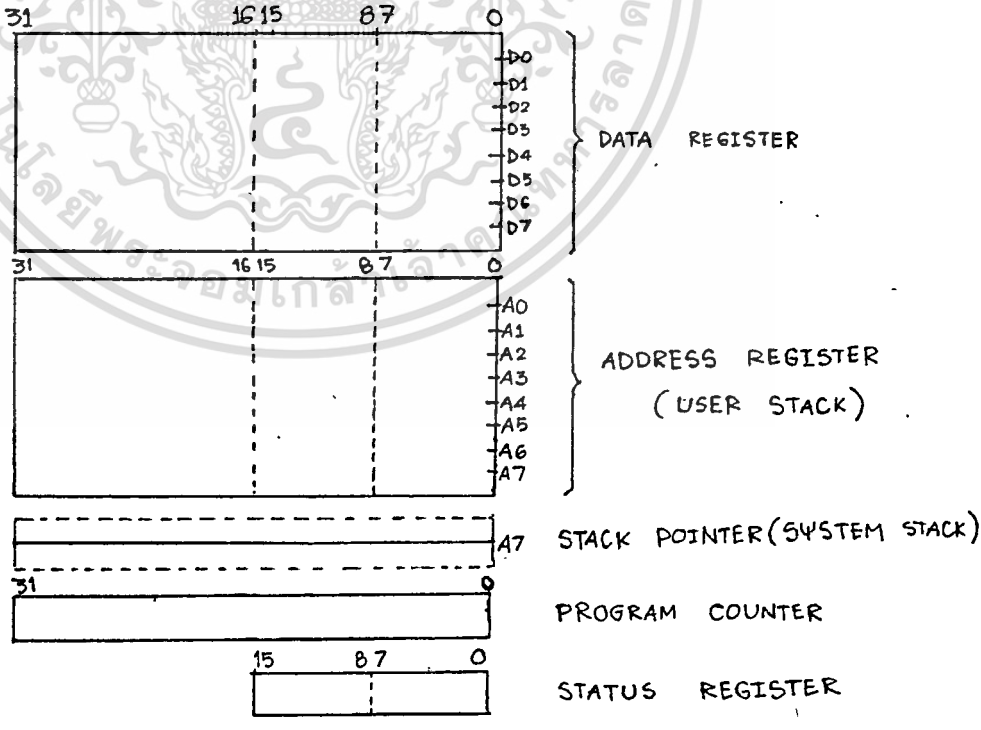
68000 ไมโครโปรเซสเซอร์

ทฤษฎีและโครงสร้างของ 68000 ไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์เบอร์ 68000 นี้จะประกอบด้วย

- ดาต้ารีจีสเตอร์และแอดเดรสรีจีสเตอร์ (Data Register and Address Register) ขนาด 32 บิท
- การกำหนดแอดเดรสโดยตรง (Direct Addressing) ขนาด 16 เมกกะไบท์ (Mbyte)
- ชุดคำสั่งที่มีประสิทธิภาพถึง 56 ชนิด
- การกระทำข้อมูลหลัก 5 ชนิด (Operation on Five Main Data Types)
- แอดเดรสซิ่งโหมด (Addressing Mode) 14 ชนิด
- การติดต่อหน่วยความจำกับอินพุท/เอาท์พุท (Memory Mapped Input/Output)

2.1 รีจิสเตอร์ (Register)



รูปที่ 2.1 แสดงรีจิสเตอร์ของ MC 68000

ไมโครโปรเซสเซอร์เบอร์ 68000 มีรีจิสเตอร์ขนาด 32 บิททั้งหมด 17 ตัว ทั้ง 17 ตัว สามารถใช้เป็นอินเด็กซ์รีจิสเตอร์ (Index Registor) ใน 17 ตัวประกอบด้วย

- ดาต้ารีจิสเตอร์ 8 ตัว
- แอดเดรสรีจิสเตอร์ 7 ตัว หรือเรียกว่าสแต็คของผู้ใช้ (User Stack)
- สแต็คพอยน์เตอร์ (Stack Pointer) 2 ตัว หรือเรียกว่าแสต็คของระบบ (System Stack)

นอกจากนี้ยังมีโปรแกรมเคาน์เตอร์ (Program Counter) ขนาด 32 บิทอีก 1 ตัว โดยที่ 24 บิทล่างเท่านั้นทำหน้าที่ส่งสัญญาณออกมา และยังมีรีจิสเตอร์สถานะ (Status Registor) ขนาด 16 บิทอีก 1 ตัว

2.1.1 ดาต้ารีจิสเตอร์ (D0 - D7)

ไมโครโปรเซสเซอร์เบอร์ 68000 มีดาต้ารีจิสเตอร์ขนาด 32 บิท 8 ตัว แต่สามารถที่จะใช้งานได้ทั้งแบบไบท์ (Byte : 8 บิท) แบบเวิร์ด (Word : 16 บิท) หรือแบบลองเวิร์ด (Long Word : 32 บิท)ก็ได้

ในการใช้งานของดาต้ารีจิสเตอร์ ถ้าไม่ถึง 32 บิท บิทที่ไม่เกี่ยวข้องจะไม่มีการเปลี่ยนแปลง ซึ่งแสดงถึงการชิพท์ (Shift)แบบตัวเลข (Arithmatic)ขนาด 8 บิทดาต้ารีจิสเตอร์จะใช้เป็นตัวรับส่งข้อมูล และยังสามารถใช้เป็นอินเด็กซ์รีจิสเตอร์(Index Registor) ได้ด้วย

2.1.2 แอดเดรสรีจิสเตอร์ (A0 - A6)

ไมโครโปรเซสเซอร์เบอร์ 68000 มีแอดเดรสรีจิสเตอร์ขนาด 32 บิท 7 ตัว สามารถใช้งานได้แบบเวิร์ดหรือลองเวิร์ด แต่ไม่สามารถใช้งานแบบไบท์ได้ ในการรับข้อมูลขนาด 16 บิท บิทที่ 16 - 31 จะมีค่าตามบิทเครื่องหมาย (Sign bit : บิทที่ 15) ทั้งนี้จะมีประโยชน์ในการบวกหรือลบแอดเดรส

การที่ไมโครโปรเซสเซอร์เบอร์ 68000 มีดาต้ารีจิสเตอร์และแอดเดรสรีจิสเตอร์ขนาด 32 บิทนี้ เป็นลักษณะพิเศษคือ ไม่มีแอคคิวมูเลเตอร์รีจิสเตอร์ (Accumulator Register)ผิดกับไมโครโปรเซสเซอร์ 16 บิทตัวอื่นๆเช่น 8086,Z-80 เพราะไมโครโปรเซสเซอร์เบอร์ 68000 สามารถใช้รีจิสเตอร์ตัวไหนเป็นแอคคิวมูเลเตอร์ก็ได้ ไม่มีการเจาะจงรีจิสเตอร์ตัวใดโดยเฉพาะ

2.1.3 สแตคพอยน์เตอร์ (A7)

ไมโครโปรเซสเซอร์เบอร์ 68000 มีสแตคพอยน์เตอร์ขนาด 32 บิท 2 ตัว ใช้งานได้ทั้งแบบเวิร์ดและลองเวิร์ด การที่มีสแตคพอยน์เตอร์ 2 ตัว ก็เพราะว่าไมโครโปรเซสเซอร์เบอร์ 68000 นี้ ทำงานได้ 2 โหมดด้วยกันคือ ซุปเปอร์ไวเซอร์โหมด (Suppervisor Mode) และยูสเซอร์โหมด (User Mode) โดยที่ขณะเวลาเดียวกันไมโครโปรเซสเซอร์เบอร์ 68000 จะทำงานได้เพียงโหมดเดียว จึงมีเพียงพอยน์เตอร์เดียวที่ทำงานขณะหนึ่งๆ

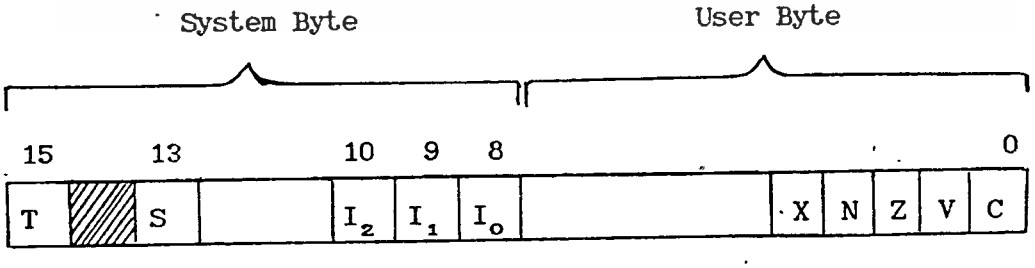
2.1.4 โปรแกรมเคาน์เตอร์ (Program Counter)

ไมโครโปรเซสเซอร์เบอร์ 68000 มีโปรแกรมเคาน์เตอร์ขนาด 32 บิท แต่จะใช้เพียง 24 บิท ซึ่งจะใช้หน่วยความจำแอดเดรส (Address Memory) ได้ขนาด 16 เมกกะไบท์ เพราะว่าโครงสร้างของไมโครโปรเซสเซอร์เบอร์ 68000 มี แอดเดรสบัส (Address Bus) 24 เส้น จึงให้หน่วยความจำแอดเดรสได้เท่ากับ 2^{24} ซึ่งประมาณ 16 เมกกะไบท์

เนื่องจากไมโครโปรเซสเซอร์เบอร์ 68000 เป็นไมโครโปรเซสเซอร์ขนาด 16 บิท ดังนั้นข้อมูลพื้นฐานจะมีขนาด 16 บิทแต่สามารถเรียกใช้แบบ 8 บิทและ 32 บิทได้ ในการเรียกใช้ข้อมูลขนาด 16 หรือ 32 บิทนี้ แอดเดรสจะต้องเป็นเลขคู่ ลักษณะเช่นนี้เป็นลักษณะทั่วไปของไมโครโปรเซสเซอร์ขนาด 16 บิท ซึ่งมีผลทำให้ค่าในโปรแกรมเคาน์เตอร์และสแตคพอยน์เตอร์ต้องเป็นเลขคู่ตามไปด้วย

2.1.5 สเตตัสรีจิสเตอร์ (Status Register)

ไมโครโปรเซสเซอร์เบอร์ 68000 ใช้สเตตัสรีจิสเตอร์ขนาด 16 บิท แบ่งเป็น 2 ส่วนคือ ไบท์ของผู้ใช้ (User Byte) กับไบท์ของระบบ (System Byte)



รูป 2.2 แสดงแต่ละบิทของสเตตัสรีจิสเตอร์

- ไบท์ของผู้ใช้

บิททด (Carry : C) ทำหน้าที่เหมือนกับแคร์รี่ในไมโครโปรเซสเซอร์ขนาด 8 บิท ทั่วๆไปคือเป็น "1" เมื่อมีตัวทดจากการบวกหรือมีตัวยืมจากการลบ นอกจากนี้แล้วคำสั่งโรเต็ต (Rotate) และชิฟท์ (Shift) ยังมีผลกับบิทบิทนี้

บิทโอเวอร์โฟลว์ (Overflow : V) บิทนี้ใช้ในการคำนวณเลขที่มีเครื่องหมาย ค่าที่ได้เกิดจากการเอ็กซ์คลูซิฟว์ออร์ (Exclusive OR) ระหว่างแคร์รี่แฟลก (Carry Flag) กับไซจ์นบิท การที่บิทนี้มีค่าเป็น "1" แสดงว่าขนาดคำตอบใหญ่เกินกว่าที่รีจิสเตอร์จะเก็บค่าไว้ได้

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

บิทเนกาทิฟ (Negative :N) บิทนี้จะมีค่าตามไซจ์นบิทของคำตอบ ถ้าบิทนี้มีค่าเป็น "0" แสดงว่าคำตอบเป็นเลขบวก แต่ถ้าบิทนี้มีค่าเป็น "1" แสดงว่าคำตอบเป็น เลขลบ

บิทเอ็กซ์เท็นด์ (Extend : X) บิทนี้ใช้เป็นแคร์รี่ในการคำนวณที่ต้องการความเที่ยงตรงมากๆ (Multiprecision) คำสั่งที่มีผลกับบิทนี้อย่างเช่น คำสั่งบวก คำสั่งลบ คำสั่งชิฟ บิทนี้จะมีค่าตามบิทแคร์รี่

สำหรับบิทที่เหลืออยู่ในปัจจุบันยังไม่มีการใช้งาน และมีค่าเป็น "0" เสมอ

- ไบท์ของระบบ จะแสดงถึงสภาวะของระบบ ในขณะที่ไบท์ของผู้ใช้จะเป็นเงื่อนไขต่างๆ ซึ่งเป็นผลจากโปรแกรม บิทต่างๆในไบท์ของระบบนี้จะเปลี่ยนค่าได้ เมื่ออยู่ในซุปเปอร์ไวเซอร์โหมดเท่านั้น

เอส - บิท (S - bit) เป็นบิทที่แสดงว่าไมโครโปรเซสเซอร์เบอร์ 68000 กำลังทำงานอยู่ในโหมดไหน ถ้าบิทนี้มีค่าเป็น "1" จะอยู่ในซุปเปอร์ไวเซอร์โหมด และถ้าบิทนี้เป็น "0" ก็จะอยู่ในยูสเซอร์โหมด โหมดทั้งสองนี้จะมีสแตคพอยน์เตอร์ของตัวเอง และมีบางคำสั่งที่สามารถใช้ได้ในซุปเปอร์ไวเซอร์โหมดเท่านั้น โดยทั่วไปซุปเปอร์ไวเซอร์โหมดนี้ มักถูกใช้สำหรับระบบปฏิบัติการ (Operating System) ส่วนยูสเซอร์จะใช้สำหรับโปรแกรมใช้งานต่างๆ (Application Program)

ที - บิท (T - bit) เป็นบิทที่ใช้แสดงว่า ขณะนั้นไมโครโปรเซสเซอร์เบอร์ 68000 อยู่ในเทรซโหมด (Trace Mode) หรือไม่ ซึ่งถ้าบิทนี้มีค่าเป็น "0" จะไม่อยู่ในเทรซโหมด แต่ถ้ามีค่าเป็น "1" ก็จะอยู่ในเทรซโหมด เทรซโหมดนี้จะมีลักษณะคล้ายการแทรพ (Trap) คือ การอินเทอร์รัพท์ ชนิดหนึ่ง ที่เกิดจากซอฟท์แวร์ ไม่ใช่สัญญาณอินเทอร์รัพท์ที่ส่งเข้ามาจาก

อุปกรณ์ภายนอก) และทำให้กลับไปที่ซุปเปอร์ไวเซอร์โหมด โดยอาจจะมีโปรแกรมดีบักเกอร์อยู่คอยแสดงผลของคำสั่งนั้นๆ

2.2 ขาและสัญญาณควบคุม

ไมโครโปรเซสเซอร์เบอร์ 68000 เป็นไอซี (IC) ที่มีขนาด 64 ขา ประกอบด้วยสัญญาณต่างๆดังนี้

- ดาต้าบัสและแอดเดรสบัส (Data Bus and Address Bus) เนื่องจากไมโครโปรเซสเซอร์เบอร์ 68000 เป็นไมโครโปรเซสเซอร์ขนาด 16 บิท ดังนั้นหน่วยของข้อมูลที่ใช้โดยทั่วไปจึงมีขนาด 16 บิท นั่นหมายความว่าในการรับส่งข้อมูลแต่ละครั้ง จะรับส่งได้ไม่เกิน 16 บิท ถ้าจะรับส่งเกิน 16 บิท ก็ต้องทำการรับส่งมากกว่า 1 ครั้ง ข้อมูลที่รับส่งระหว่างไมโครโปรเซสเซอร์เบอร์ 68000 กับอุปกรณ์ภายนอก (อาจจะเป็นหน่วยความจำหรืออุปกรณ์อินพุทเอาท์พุทต่างๆ) จะรับส่งผ่านดาต้าบัส (D0 - D15)

เพื่อที่จะให้อุปกรณ์ภายนอกทราบว่าไมโครโปรเซสเซอร์เบอร์ 68000 จะติดต่อกับอุปกรณ์ตัวไหน ไมโครโปรเซสเซอร์เบอร์ 68000 จะใช้สัญญาณจากแอดเดรสบัส (A1 - A23) เลือกอุปกรณ์ โดยส่งแอดเดรสของอุปกรณ์นั้นไปตามขา A1 - A23 พร้อมกับมีสัญญาณแอดเดรสสโตรบ (Address Strobe : AS) เพื่อบอกอุปกรณ์ภายนอกว่า ตอนนี้แอดเดรสบนแอดเดรสบัสเป็นแอดเดรสที่ถูกต้องแล้ว

- การรับส่งข้อมูลแบบอะซิงโครนัส (Asynchronous)

เนื่องจากไมโครโปรเซสเซอร์เบอร์ 68000 สามารถเรียกใช้ข้อมูลขนาด 1 ไบท์ได้ ทำให้อ้างแอดเดรสได้ถึง 16 เมกกะไบท์ ซึ่งต้องใช้สัญญาณแอดเดรสขนาด 24 บิท แต่ว่าไมโครโปรเซสเซอร์เบอร์ 68000 มีขาแอดเดรสเพียง 23 ขา ดังนั้นจึงมี สัญญาณ LDS (Lower Data Strobe) และ UDS (Upper Data Strobe) ช่วย โดยเมื่อสัญญาณ LDS เป็น"0" ข้อมูลจะถูกรับส่งในดาต้าบัส 8 บิทล่าง (D0 - D7) และถ้าสัญญาณ UDS เป็น "0" ข้อมูลจะถูกรับส่งในดาต้าบัส 8 บิทบน (D8 - D15) ดังนั้นถ้าสัญญาณทั้ง 2 เป็น 0 ข้อมูลทั้ง 16 บิทจะถูกรับส่งพร้อมๆกัน

ต่อมาเพื่อที่จะให้อุปกรณ์ภายนอกทราบว่าไมโครโปรเซสเซอร์เบอร์ 68000 กำลังจะรับ (Read) หรือส่ง (Write) ไมโครโปรเซสเซอร์เบอร์ 68000 ก็จะใช้สัญญาน R/W เป็นตัวบอกคือ

ถ้าสัญญาณ R/W เป็น "1" แสดงว่าไมโครโปรเซสเซอร์เบอร์ 68000 กำลังต้องการ

รับข้อมูล

และถ้าเป็น "0" แสดงว่ากำลังส่งข้อมูล

การทำงานของสัญญาน UDS , LDS และ R/W หลังจากที่เราสามารถบอกได้ว่า อุปกรณ์ไหนที่ใช้รับส่งข้อมูล ไบท์ไหนหรือทั้งสองไบท์ และจะทำการรับหรือส่งข้อมูลแล้ว สิ่งต่อไปก็คือการตอบรับของอุปกรณ์ภายนอกมายังไมโครโปรเซสเซอร์เบอร์ 68000 ว่า ส่งข้อมูลมาให้แล้วหรือรับข้อมูลไว้แล้ว ซึ่งจะใช้สัญญาณดาต้าทรานสเฟอร์แอคโนว์เลดจ์ (Data Transfer Acknowladge : DATCK) ถ้าไม่มีสัญญาณนี้ตอบรับกลับมา ไมโครโปรเซสเซอร์เบอร์ 68000 จะรอไปเรื่อยๆ วิธีนี้จะทำให้ไมโครโปรเซสเซอร์เบอร์ 68000 สามารถที่จะชลอการทำงานเมื่อติดต่อกับอุปกรณ์ที่ทำงานช้า และทำงานได้เร็วขึ้นเมื่อติดต่อกับอุปกรณ์ที่ทำงานได้เร็ว จะเห็นได้ว่าการรับส่งข้อมูลแบบนี้จะอาศัยการตอบรับ(hand shaking) ไม่ใช่กำหนดช่วงเวลาให้รับส่งให้ทันการรับส่งข้อมูลโดยอาศัยการตอบรับนี้ การรับส่งเป็นแบบอะซิงโครนัส

- สัญญาณรหัสฟังก์ชั่น (Function Code)

ไมโครโปรเซสเซอร์เบอร์ 68000 นอกจากจะมีสัญญาณตอบรับสำหรับส่งข้อมูลแล้ว ยังมีสัญญาณที่ใช้บอกเหตุการณ์ภายนอกว่าข้อมูลที่รับส่งนั้นเป็นข้อมูลอะไร เช่น โปรแกรม หรือข้อมูลของผู้ใช้ หรือข้อมูลของซุปเปอร์ไวเซอร์ สัญญาณที่ใช้บอกนี้คือ สัญญาณรหัสฟังก์ชั่น (FC0, FC1,FC2)

สัญญาณรหัสฟังก์ชั่นนี้สามารถใช้ขยายเนื้อที่หน่วยความจำจาก 16 เมกกะไบท์เป็น 64 เมกกะไบท์ได้ นอกจากนี้แล้วยังสามารถใช้สัญญาณรหัสฟังก์ชั่นจัดการเกี่ยวกับหน่วยความจำอย่างเช่น ไม่ให้โปรแกรมผู้ใช้ใช้เนื้อที่หน่วยความจำบางส่วนที่สงวนไว้สำหรับการทำงานในซุปเปอร์ไวเซอร์โหมด

- การรับส่งข้อมูลแบบซิงโครนัส (Synchronous)

นอกจากจะรับส่งข้อมูลแบบอะซิงโครนัส ซึ่งใช้การตอบรับโดยไม่ขึ้นกับความเร็วของอุปกรณ์ที่ติดต่อด้วยแล้ว ไมโครโปรเซสเซอร์เบอร์ 68000 ยังสามารถรับส่งข้อมูลแบบซิงโครนัส การรับส่งแบบซิงโครนัสนี้ไมโครโปรเซสเซอร์เบอร์ 68000 ไม่ได้รอสัญญาณ DATCK แต่ใช้สัญญาณนาฬิกาเพื่อกำหนดช่วงเวลาในการรับส่งแทน

การรับส่งแบบซิงโครนัสใช้สัญญานเอ็นเอเบิล (Enable : E) เป็นตัวส่งสัญญาณนาฬิกาดังกล่าวจากไมโครโปรเซสเซอร์เบอร์ 68000 โดยใช้ความถี่เพียง 1/10 ของสัญญาณนาฬิกาที่ไมโครโปรเซสเซอร์เบอร์ 68000 ใช้จริง และรูปสัญญาณจะมีขนาด 40/60 คือเป็น "1"

อยู่ 4 สัญญาณและเป็น "0" อยู่ 6 สัญญาณ

การรับส่งแบบซิงโครนัสนี้อุปกรณ์ภายนอกจะส่งสัญญาณ VPA (Valid Peripheral Acknowledge) มาให้ไมโครโปรเซสเซอร์เบอร์ 68000 เพื่อบอกให้เราทราบว่าต่อไปจะรับส่งข้อมูลแบบซิงโครนัส ไมโครโปรเซสเซอร์เบอร์ 68000 จะตอบรับทราบแล้ว โดยใช้สัญญาณ VMA (Valid Memory Acknowledge)

การที่ไมโครโปรเซสเซอร์เบอร์ 68000 สามารถที่จะรับส่งข้อมูลแบบซิงโครนัส ทำให้ไมโครโปรเซสเซอร์เบอร์ 68000 สามารถใช้อุปกรณ์สนับสนุนที่เคยใช้กับไมโครโปรเซสเซอร์เบอร์ 68000 ได้ ไมโครโปรเซสเซอร์เบอร์ 6800 เป็นไมโครโปรเซสเซอร์ขนาด 8 บิท ของโมโตโรลาเช่นเดียวกัน และจัดว่าเป็นไมโครโปรเซสเซอร์ตระกูลหนึ่งที่มีชื่อเสียงและอุปกรณ์ประกอบจำนวนมาก ซึ่งอุปกรณ์เหล่านั้นส่วนมากก็มีการรับส่งข้อมูลแบบซิงโครนัส ด้วยวิธีนี้ไมโครโปรเซสเซอร์เบอร์ 68000 ก็มีอุปกรณ์ประกอบเพิ่มขึ้นอีกมาก ซึ่งไมโครโปรเซสเซอร์ขนาด 16 บิทอื่นๆก็ใช้วิธีนี้อย่างเช่น 8086 ของอินเทล สามารถใช้อุปกรณ์ร่วมกับ 8080 อันเป็นไมโครโปรเซสเซอร์ 8 บิทที่มีชื่อเสียงของบริษัทอินเทลเองได้

- สัญญาณควบคุมบัสและระบบ

สัญญาณ BR (Bus Request),BG(Bus Grant),BGACK(Bus Grant Acknowage) เป็นสัญญาณควบคุมบัส สำหรับการทำ DMA(Direct Memory Access) สัญญาณรีเซ็ท (RESET), BERR(Bus Error) และ HALT เป็นสัญญาณสองทิศทาง นอกจากอุปกรณ์ภายนอกจะส่งสัญญาณเข้ามาแล้ว ไมโครโปรเซสเวอร์เบอร์ 68000 ยังสามารถส่งสัญญาณนี้ไปควบคุมอุปกรณ์ภายนอก โดยตัวไมโครโปรเซสเซอร์เบอร์ 68000 ไม่ต้องถูกฮอลท์ (Halt) หรือรีเซ็ทจริงๆอีกด้วย สัญญาณ BERR เป็นสัญญาณที่ใช้ประกอบกับ DTACK โดยเมื่ออุปกรณ์ภายนอกเกิดความผิดพลาดไม่สามารถที่จะส่งสัญญาณ DTACK ได้ สัญญาณ BERR ก็จะถูกส่งเข้ามาบอกไมโครโปรเซสเซอร์เบอร์ 68000 (อาจส่งมาจากอุปกรณ์อีกตัวหนึ่ง ซึ่งทำหน้าที่ตรวจสอบความผิดพลาด) ว่ามีข้อผิดพลาด (Error)เกิดขึ้น

- สัญญาณอินเทอร์รัพท์ (Interrupt)

ประกอบด้วยสัญญาณ 3 ตัวคือ สัญญาณ IP0,IP1 และ IP2 ซึ่งรวมกันแล้วจะแสดงถึงระดับไพรออริตี้ (Priority) หรือระดับความสำคัญของการอินเทอร์รัพท์ การที่สัญญานอินเทอร์รัพท์จะทำการอินเทอร์รัพท์ได้นั้น จะต้องมีระดับไพรออริตี้สูงกว่าหรือเท่ากับที่มาส์ค (Mask) ไว้ในสเตตัสรีจิสเตอร์ ถ้าสามารถอินเทอร์รัพท์เข้ามาได้ ไมโครโปรเซสเซอร์เบอร์ 68000 ก็

จะไปประมวลผลรูทีน (Routine) ต่างๆตามเวคเตอร์ (Vector คือ แอดเดรสของรูทีนต่างๆ)

2.3 เอ็กซ์เซ็พท์ชั่น (Exception)

เอ็กซ์เซ็พท์ชั่นก็คือการอินเทอร์รัพท์ ที่บริษัทโมโตโรลาใช้ชื่อนี้เพราะไมโครโปรเซสเซอร์เบอร์ 68000 สามารถทำการอินเทอร์รัพท์ได้มากชนิดกว่าไมโครโปรเซสเซอร์เบอร์อื่นมาก

วิธีการจัดการกับเอ็กซ์เซ็พชั่นนั้นไมโครโปรเซสเซอร์เบอร์ 68000 จะใช้ตารางซึ่งเก็บค่าเวคเตอร์ที่จะจัดการกับการอินเทอร์รัพท์ชนิดต่างๆ เมื่อไมโครโปรเซสเซอร์เบอร์ 68000 รู้ว่าเป็นอินเทอร์รัพท์ชนิดไหน ก็จะสามารถไปทำรูทีนนั้นตามค่าเวคเตอร์บนตาราง ตารางที่ใช้เก็บค่าเวคเตอร์นี้มีชื่อว่าตารางเอ็กซ์เซ็พชั่นเวคเตอร์ (Exception Vector Table) ไมโครโปรเซสเซอร์ขนาด 16 บิทชนิดอื่น เช่น 8086 และ 8080 ก็ใช้วิธีเดียวกันนี้ ต่างกันตรงที่ว่าไมโครโปรเซสเซอร์เบอร์ 68000 มีตารางที่ใหญ่กว่ามาก นอกจากนี้ยังสามารถแบ่งการอินเทอร์รัพท์จากภายนอกที่เข้ามาได้ถึง 7 ระดับ

2.3.1 โหมดของการทำงานกับการเกิดเอ็กซ์เซ็พชั่น

ดังที่กล่าวมาแล้วไมโครโปรเซสเซอร์เบอร์ 68000 แบ่งการทำงานออกเป็น 2 โหมด คือ ซุปเปอร์ไวเซอร์โหมดกับยูสเซอร์โหมด เมื่อไมโครโปรเซสเซอร์เบอร์ 68000 ถูกรีเซ็ท มันจะเริ่มทำงานในซุปเปอร์ไวเซอร์โหมด จนกว่าจะมีคำสั่งที่เปลี่ยนค่าบิทในสเตตัสรีจิสเตอร์ให้เป็น "0" จึงจะไปทำงานในยูสเซอร์โหมดต่อไป

เมื่อไมโครโปรเซสเซอร์เบอร์ 68000 ทำงานอยู่ในยูสเซอร์โหมด มีแต่การเอ็กซ์เซ็พท์ชั่นเท่านั้นที่จะทำให้ไมโครโปรเซสเซอร์เบอร์ 68000 กลับไปทำงานในซุปเปอร์ไวเซอร์โหมดได้

2.3.2 ชนิดของการเอ็กซ์เซ็พท์ชั่น

แบ่งออกเป็น ชนิดใหญ่ๆ ดังนี้ คือ

1. การเอ็กซ์เซ็พท์ชั่นที่เกิดจากภายในซึ่งมีผลเกิดจากคำสั่งบางคำสั่ง เช่นคำสั่งแทรพ (Trap เป็นซอฟท์แวร์อินเทอร์รัพท์ เทียบได้กับคำสั่ง Restart ของ 8080) หรือเกิดจากความผิดพลาดบางอย่างที่พบภายใน ได้แก่

- แอดเดรสผิด (Address Error) ไมโครโปรเซสเซอร์เบอร์ 68000 พยายามที่จะเรียกใช้ข้อมูลขนาด 16 หรือ 32 บิทหรือคำสั่งที่มีแอดเดรสเป็นเลขคี่
- พยายามที่จะทำคำสั่งที่ไม่อนุญาตให้ใช้ในยูสเซอร์โหมด
- คำสั่งผิดหรือไม่มีในชุดคำสั่ง

ไมโครโปรเซสเซอร์เบอร์ 68000 นอกจากจะมีคำสั่งแทรพโดยตรงแล้ว ยังมีคำสั่งอื่นอีกที่จะทำให้เกิดการแทรพเมื่อมีเงื่อนไขสอดคล้องกับที่ต้องการ อย่างเช่นให้แทรพเมื่อเกิดโอเวอร์โฟลว์ ได้แก่คำสั่ง TRAPV

นอกจากนี้อาจเกิดจากฟังก์ชั่นเทรซ (Trace) โดยที่ฟังก์ชั่นเทรซจะเริ่มทำเมื่อ ที-บิทในสเตตัสรีจิสเตอร์เป็น "1" โดยหลังจากทำคำสั่งหนึ่งในยูสเซอร์โหมดแล้ว จะเกิดการเอ็กซ์เซ็พท์ชั่นขึ้นทันที ช่วยให้สามารถทำซิงเกิลสเต็ป (Single Step) ได้

2. การเอ็กซ์เซ็พท์ชั่นที่เกิดจากภายนอก ได้แก่สัญญาณที่อุปกรณ์ภายนอกส่งเข้ามาอินเทอร์รัพท์ คือ

- สัญญาณ BERR
- สัญญาณ RESET
- สัญญาณอินเทอร์รัพท์ (IP0, IP1, IP2)

2.3.3 ระดับความสำคัญของการเอ็กซ์เซ็พท์ชั่น

เมื่อเกิดการเอ็กซ์เซ็พท์ชั่นขึ้นหลายอย่างพร้อมๆกัน ไมโครโปรเซสเซอร์เบอร์ 68000 จะจัดการกับเอ็กซ์เซ็พท์ชั่นที่มีระดับสูงสุดก่อน และในระหว่างการจัดการนั้นจะยอมให้มีการเอ็กซ์เซ็พท์ชั่นที่สูงกว่ามาเอ็กซ์เซ็พท์ชั่นได้เท่านั้น ระดับความสำคัญของการเอ็กซ์เซ็พท์ชั่นสามารถแบ่งออกเป็นชุดใหญ่ๆ ได้ทั้งหมด 3 ชุดคือ

- ชุดที่มีระดับความสำคัญสูงที่สุด จะหยุดการทำคำสั่งทันทีเมื่อเกิดการเอ็กซ์เซ็พท์ชั่นในชุดนี้ขึ้น โดยไม่ต้องรอให้ทำคำสั่งนั้นเสร็จ
- ชุดที่มีระดับความสำคัญรองลงมา จะรอให้ทำจนจบคำสั่งก่อนจึงเริ่มทำการเอ็กซ์เซ็พท์ชั่น ในชุดนี้จะรวมถึงการอินเทอร์รัพท์ซึ่งแบ่งระดับความสำคัญออกอีก 7 ระดับด้วย
- ชุดที่มีระดับความสำคัญต่ำที่สุด จะรอให้จบคำสั่งก่อนเช่นเดียวกับชุดที่สอง

2.3.4 ตารางเอ็กซ์เซ็พท์ชั่นเวคเตอร์

ศูนย์กลางของการจัดการเอ็กซ์เซ็พท์ชั่นอยู่ที่ตารางนี้ ซึ่งมีขนาด 1024 ไบท์ อยู่ในตำแหน่งหน่วยความจำที่ 000000H - 0003FFH

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

- ส่วนที่กำหนดไว้แล้ว สำหรับเอ็กซ์เซ็พท์ชั่นชนิดต่างๆ ซึ่งได้แก่ 64 แอดเดรส

(เวคเตอร์)แรก

- และส่วนที่เหลืออีก 292 แอดเดรส (เวคเตอร์) นั้นผู้ใช้สามารถกำหนดได้เอง นอกจากนี้แล้วจะเห็นได้ว่ามีบางแอดเดรสที่ถูกสำรองไว้โดยบริษัทโมโตโรลาผู้ผลิตเอง อุปกรณ์ภายนอกสามารถใช้แอดเดรสในเวคเตอร์ของผู้ใช้ได้ โดยเมื่อส่งสัญญานอินเทอร์รัพท์เข้ามาแล้วไมโครโปรเซสเซอร์เบอร์ 68000 จะตอบโดยส่งอินเทอร์รัพท์แอคโนว์เลดจ์ (ใช้สัญญานรหัสฟังก์ชั่นและมีสัญญาณ A1 - A3 แสดงระดับความสำคัญที่อินเทอร์รัพท์เข้ามา) อุปกรณ์ก็จะส่งสัญญาณ DTACK พร้อมกับเลขที่เวคเตอร์มาตามสาย D0 - D7 แต่ถ้าไม่ต้องการกำหนดเลขที่เวคเตอร์เองก็ส่งสัญญาณ VPA มา ไมโครโปรเซสเซอร์เบอร์ 68000 ก็จะทำการประมวลผลรูทีนตามที่กำหนดไว้แล้วสำหรับอินเทอร์รัพท์ ระดับต่างๆ การที่ผู้ใช้สามารถกำหนดเวคเตอร์ได้เองนี้ ทำให้ไมโครโปรเซสเซอร์เบอร์ 68000 สามารถทำการเอ็กซ์เซ็พท์ชั่นได้มากกว่า 200 ชนิด

2.4 โหมดในการแอดเดรส (Addressing Mode)

คำสั่งของไมโครโปรเซสเซอร์เบอร์ 68000 ส่วนใหญ่จะมีโอเปอร์แรนด์ (Operand) ตั้งแต่ 1 ตัวขึ้นไป โดยที่โอเปอร์แรนด์ตัวหนึ่งๆอาจจะอยู่ภายในซีพียู (คือในรีจิสเตอร์) หรือภายนอกซีพียู (คือในหน่วยความจำ)

โหมดในการแอดเดรสจะเป็นตัวกำหนดว่า ซีพียูจะจัดการกับแอดเดรสที่ใช้ในการประมวลผล (Effective Address) ของโอเปอร์แรนด์ที่อยู่ในรีจิสเตอร์หรือหน่วยความจำอย่างไร ซึ่งไมโครโปรเซสเซอร์เบอร์ 68000 มีวิธีการแอดเดรสได้ทั้งหมด 6 ชนิดคือ

1) การแอดเดรสแบบรีจิสเตอร์ไดเร็คท์ (Registor Direct Addressing) เป็นการกำหนดรีจิสเตอร์ที่จะใช้งานโดยตรงได้แก่

- ดาต้ารีจิสเตอร์ไดเร็คท์ (Data Registor Direct)

โหมดในการแอดเดรสโหมดนี้ ดาต้ารีจิสเตอร์จะเป็นโอเปอร์แรนด์ ซึ่งระบุโดยใช้นีโมนิค"Dn" โดยที่"D"หมายถึง โอเปอร์แรนด์ที่เป็นดาต้ารีจิสเตอร์ และ"n"หมายถึงหมายเลขรีจิสเตอร์ตั้งแต่ 0 - 7

- แอดเดรสรีจิสเตอร์ไดเร็คท์ (Address Register Direct)

คล้ายกับดาต้ารีจิสเตอร์ไดเร็คท์ ยกเว้นว่ารีจิสเตอร์ที่ใช้เป็นแอดเดรสรีจิสเตอร์ เราระบุโหมดนี้โดยใช้นีโมนิค "An" โดยที่ "A" หมายถึงแอดเดรสรีจิสเตอร์ และ "n" หมายถึงหมายเลขรีจิสเตอร์ตั้งแต่ 0 - 7

2) การแอดเดรสแบบรีจิสเตอร์อินไดเร็คท์ (Register Indirect Addressing) วิธีการแอดเดรสชนิดนี้ ค่าของแอดเดรสที่ต้องการจะอยู่ในแอดเดรส รีจิสเตอร์ ซึ่งอาจจะมีการบวก หรือลบด้วยค่าคงที่ หรือค่าในอินเด็กซ์รีจิสเตอร์ (Index Register จะเป็นตาต้ารีจิสเตอร์หรือแอดเดรสรีจิสเตอร์ก็ได้) ได้แก่

- รีจิสเตอร์อินไดเร็คท์ (Register Indirect)

การแอดเดรสชนิดนี้ โอเปอร์แรนด์จะอยู่ในหน่วยความจำ แอดเดรสรีจิสเตอร์จะเก็บแอดเดรสของโอเปอร์แรนด์ไว้ โดยการระบุแอดเดรสรีจิสเตอร์ภายในวงเล็บเช่น (A3)

- พรีดีครีเมนท์รีจิสเตอร์อินไดเร็คท์ (Predecrement Register Indirect)

การแอดเดรสโหมดนี้ แอดเดรสรีจิสเตอร์จะเก็บค่าแอดเดรสไว้ในหน่วยความจำ อย่างไรก็ตามก่อนการกำหนดค่าแอดเดรสของโอเปอร์แรนด์ ซีพียูจะลบค่าออกจากแอดเดรสรีจิสเตอร์ คือค่าแอดเดรสที่อยู่ในหน่วยความจำ ค่าของการลบจะขึ้นอยู่กับขนาดของการโอเปอร์เรชั่น (Operation) คือลบด้วย 1 สำหรับไบท์โอเปอร์เรชั่น, 2 สำหรับเวิร์ดโอเปอร์เรชั่น และ 4 สำหรับลองเวิร์ดโอเปอร์เรชั่น

หลังจากการลบ ซีพียูจะเก็บค่าใหม่ที่ได้นี้ไว้ในแอดเดรสรีจิสเตอร์ และใช้ค่านี้เป็นแอดเดรส ที่ใช้ในการประมวลผล ของโอเปอร์แรนด์ ในแอสเซมเบลอร์สเตทเมนท์ (Assembler Statement) การแอดเดรสโหมดนี้จะระบุเครื่องหมาย "-" ก่อนวงเล็บ เช่น -(A5)

- โพสท์อินครีเมนท์รีจิสเตอร์อินไดเร็คท์ (Postincrement Register Indirect)

ในการแอดเดรสโหมดนี้คล้ายกับพรีดีครีเมนท์โหมด แอดเดรสรีจิสเตอร์จะมีแอดเดรสอยู่ในหน่วยความจำ และซีพียูจะเปลี่ยนแปลงค่าแอดเดรสรีจิสเตอร์ตามขนาดของการโอเปอร์เรชั่น

อย่างไรก็ตาม ในกรณีนี้ซีพียูจะใช้ค่าที่มีอยู่ในแอดเดรสรีจิสเตอร์ขณะนั้นเป็นแอดเดรสที่ใช้ในการประมวลผลของโอเปอร์แรนด์ หลังจากเก็บค่านี้แล้ว ซีพียูจะเพิ่มขนาดของการโอเปอร์เรชั่นเข้าไว้ในแอดเดรสรีจิสเตอร์ รูปแบบของการแอสเซมเบลอร์ (Assembler Syntax) สำหรับโหมดนี้จะมีแอดเดรสอยู่ภายในวงเล็บ และตามด้วยเครื่องหมาย "+" เช่น (A5)+

- รีจิสเตอร์อินไดเร็คท์กับดิสเพลซเมนท์ (Register Indirect with Displacement)

ในการแอดเดรสโหมดนี้ แอดเดรสที่ใช้ในการประมวลผลของโอเปอร์แรนด์ จะเป็นผลบวกของดิสเพลซเมนท์ 16 บิทที่มีเครื่องหมายกับค่าที่อยู่ในแอดเดรสรีจิสเตอร์ ก่อนที่ซีพียูจะบวกค่าดิสเพลซเมนท์กับค่าในแอดเดรสรีจิสเตอร์ มันจะทำการขยายเครื่องหมาย (Sign Extend) ของดิสเพลซเมนท์จากบิทที่ 15 ไปเป็นบิทที่ 16 ถึง 31 ซึ่งจะมีทั้งดิสเพลซเมนท์ที่เป็นบวกและลบ (Positive & Negative)

ในโหมดนี้จะคล้ายกับรีจิสเตอร์อินไดเร็คท์โหมดอื่น รูปแบบของการแอสเซมเบลอร์จะใช้แอดเดรสรีจิสเตอร์อยู่ภายในวงเล็บ ค่าดิสเพลซเมนท์จะอยู่ก่อนแอดเดรสรีจิสเตอร์ เช่น 10(A1) โดยที่ค่าดิสเพลซเมนท์นี้จะเป็นค่าคงที่ ในขณะที่ค่าของแอดเดรสอาจเปลี่ยนไปตามการประมวลผลของโปรแกรม

3) **การแอดเดรสแบบอิมไพลด์รีจิสเตอร์ (Implied Register Addressing)** หมายถึงการใช้สแตคพอยน์เตอร์, สเตตัสรีจิสเตอร์หรือโปรแกรมเคาน์เตอร์ ซึ่งจำเป็นต้องใช้อยู่แล้วสำหรับคำสั่งนั้นๆ

4) **แอบโซลูทแอดเดรสซิ่ง (Absolute Addressing)** เป็นการกำหนดแอดเดรสโดยตรงในคำสั่ง โดยที่แอดเดรสอาจจะเป็น 16 บิทหรือ 32 บิท ถ้าเป็น 16 บิทแล้วซีพียูจะทำการขยายเครื่องหมายก่อนใช้มัน รูปแบบของการแอสเซมเบลอร์สำหรับโหมดนี้ ค่าแอดเดรสจะตามด้วย .L หรือ .W ถ้ามันรู้ค่าของแอดเดรสจะสามารถตัดสินใจได้ว่าแอดเดรสจะเป็น 16 หรือ 32 บิท

5) **การแอดเดรสแบบโปรแกรมเคาน์เตอร์รีเลทีฟ (Program Counter Relative Addressing)** เป็นการแอดเดรสโดยเทียบกับโปรแกรมเคาน์เตอร์ โดยจะทำการบวกค่าในโปรแกรมเคาน์เตอร์ด้วยค่าในอินเด็กซ์รีจิสเตอร์ หรือค่าคงที่ ได้แก่

- โปรแกรมเคาน์เตอร์อินไดเร็คท์กับดิสเพลซเมนท์ (Program Counter Indirect with Displacement)

ในโหมดการแอดเดรสนี้ คล้ายกับโหมดของแอดเดรสรีจิสเตอร์อินไดเร็คท์กับดิสเพลซเมนท์ ยกเว้นว่าค่าแอดเดรสที่ใช้ในการประมวลผลจะเป็นค่าดิสเพลซเมนท์จากค่าที่อยู่ในโปรแกรมเคาน์เตอร์ขณะนั้นแทน ค่าของโปรแกรมเคาน์เตอร์ที่ใช้จะเป็นแอดเดรสของโอเปอร์แรนด์ในส่วนของคำสั่งนั้น ค่าดิสเพลซเมนท์ในโหมดนี้จะเป็น 16 บิทที่คิดเครื่องหมายด้วย โดยจะปิดดิสเพลซเมนท์และโปรแกรมเคาน์เตอร์นี้ไว้ในวงเล็บ เช่น 10(PC)

6) **อิมมีเดียทดาต้าแอดเดรสซิ่ง (Immediate Data Addressing)** เป็นการกำ

หนดข้อมูลโดยตรง อาจจะอยู่ในออพโค้ด(Opcode)เลย หรืออยู่ต่อจากออพโค้ดก็ได้ โดยระบุตัวอักษร "#" ไว้หน้าโอเปอร์แรนด์ เช่น #123 และอาจต่อท้ายโอเปอร์แรนด์ด้วยตัวบอกขนาดคือ .B .W .L ถ้าหากไม่ใส่ตัวแอสเซมเบลอร์ จะเลือกขนาดตามขนาดของค่าเอง

2.5 ชุดคำสั่งของ 68000

MC 68000 มีคำสั่งพื้นฐานอยู่ 56 คำสั่งด้วยกัน แต่เมื่อรวมกับการแอดเดรสอีกหลายชนิด ทำให้ชุดคำสั่งมีมากกว่า 300 คำสั่ง เนื่องจากไมโครโปรเซสเซอร์เบอร์ 68000 ไม่มีคำสั่งเฉพาะสำหรับอุปกรณ์อินพุท/เอาท์พุท แต่คำสั่งทุกคำสั่งที่อ้างถึงหน่วยความจำก็ใช้เป็นคำสั่งอินพุท/เอาท์พุทได้ เพราะไมโครโปรเซสเซอร์เบอร์ 68000 จะถือว่าอุปกรณ์อินพุท/เอาท์พุทเสมือนหน่วยความจำตำแหน่งหนึ่งที่อ่านหรือเขียนข้อมูลได้เช่นเดียวกับหน่วยความจำทั่วไป ซึ่งวิธีการนี้เรียกว่า " Memory Mapped I/O"

รูปแบบของคำสั่งของไมโครโปรเซสเซอร์เบอร์ 68000 จะเริ่มด้วยออพโค้ดขนาด 2 ไบท์เสมอ ส่วนคำสั่งทั้งหมดอาจจะยาวได้ตั้งแต่ 2 ไบท์ถึง 10 ไบท์

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

ไมโครโปรเซสเซอร์เบอร์ 68000 ไม่มีคำสั่งที่ใช้ย้ายข้อมูลเป็นบล็อค (Block) อย่างของ Z 8000 หรือ 8086 แต่ไมโครโปรเซสเซอร์เบอร์ 68000 มีคำสั่งพื้นฐานที่ทำให้เขียนโปรแกรมที่ย้ายข้อมูลเป็นบล็อคได้ง่ายๆ

ไมโครโปรเซสเซอร์เบอร์ 68000 มีคำสั่งสำหรับการคูณและการหารซึ่ง Z 8000 ก็มี โดย Z 8000 สามารถคูณและหารข้อมูลได้ เฉพาะแบบที่ไม่มีเครื่องหมายเท่านั้น ส่วนไมโครโปรเซสเซอร์เบอร์ 68000 สามารถคูณและหารแบบมีหรือไม่มีเครื่องหมายก็ได้ แต่ Z 8000 สามารถคูณและหารขนาด 32 บิทได้ ส่วนไมโครโปรเซสเซอร์เบอร์ 68000 คูณและหารได้แค่แบบ 16 บิทเท่านั้น

คำสั่งบางคำสั่งที่น่าสนใจของไมโครโปรเซสเซอร์เบอร์ 68000 ก็คือคำสั่ง LINK และ UNLK คำสั่งพวกนี้จะใช้จองที่สำหรับจัดการเกี่ยวกับการส่งผ่านข้อมูลระหว่างโปรแกรมย่อย (Subroutine) ต่างๆ

บทที่ 3

การพัฒนาด้านฮาร์ดแวร์ (Hardware Development)

3.1 การสร้างระบบไมโครโปรเซสเซอร์

ในระบบไมโครคอมพิวเตอร์จะประกอบด้วยส่วนต่างๆ 3 ส่วนคือ

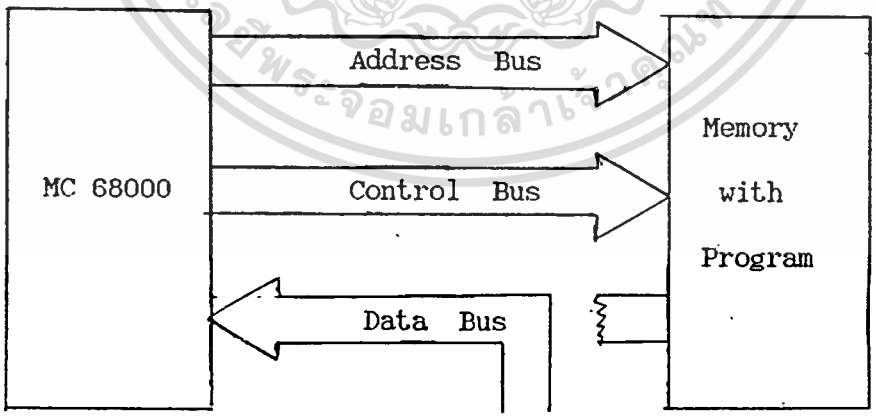
1. ไมโครโปรเซสเซอร์ มีหน้าที่ประมวลผล

2. หน่วยความจำ มีหน้าที่เก็บข้อมูลและเก็บโปรแกรมสำหรับการทำงาน

3. ส่วนติดต่อกับอุปกรณ์ภายนอก อินพุท/เอาท์พุท มีหน้าที่แปลสัญญาณต่างๆ จุดประสงค์เพื่อให้คอมพิวเตอร์สามารถติดต่อกับอุปกรณ์ภายนอกได้

สำหรับระบบไมโครโปรเซสเซอร์ตามความหมายในที่นี้ก็คือ ระบบที่ประกอบด้วยซีพียูอย่างเดียวล้วนๆ

การพิจารณาระบบไมโครคอมพิวเตอร์ ควรเริ่มจากระบบไมโครโปรเซสเซอร์ก่อน การที่จะให้ระบบไมโครโปรเซสเซอร์นี้ทำงาน ก็ควรจะให้มันทำคำสั่งที่ไม่ต้องทำอะไรเลย (Do-Nothing Instruction) นั่นก็คือทำคำสั่ง NIL ซึ่งเป็นคำสั่งที่มีความยาว 16 บิทหรือ 1 เวิร์ด โดยตัวซีพียูจะทำคำสั่งนี้ไปเรื่อยๆจนครบ 16 เมกกะไบท์ แล้วจะวนกลับมาทำอีก วิธีการที่จะให้ซีพียูทำคำสั่ง NIL ตลอดก็คือ ตัดขาดาต้าออกแล้วนำคำสั่ง NIL ใส่เข้าไปแทนดังรูปที่ 3.1



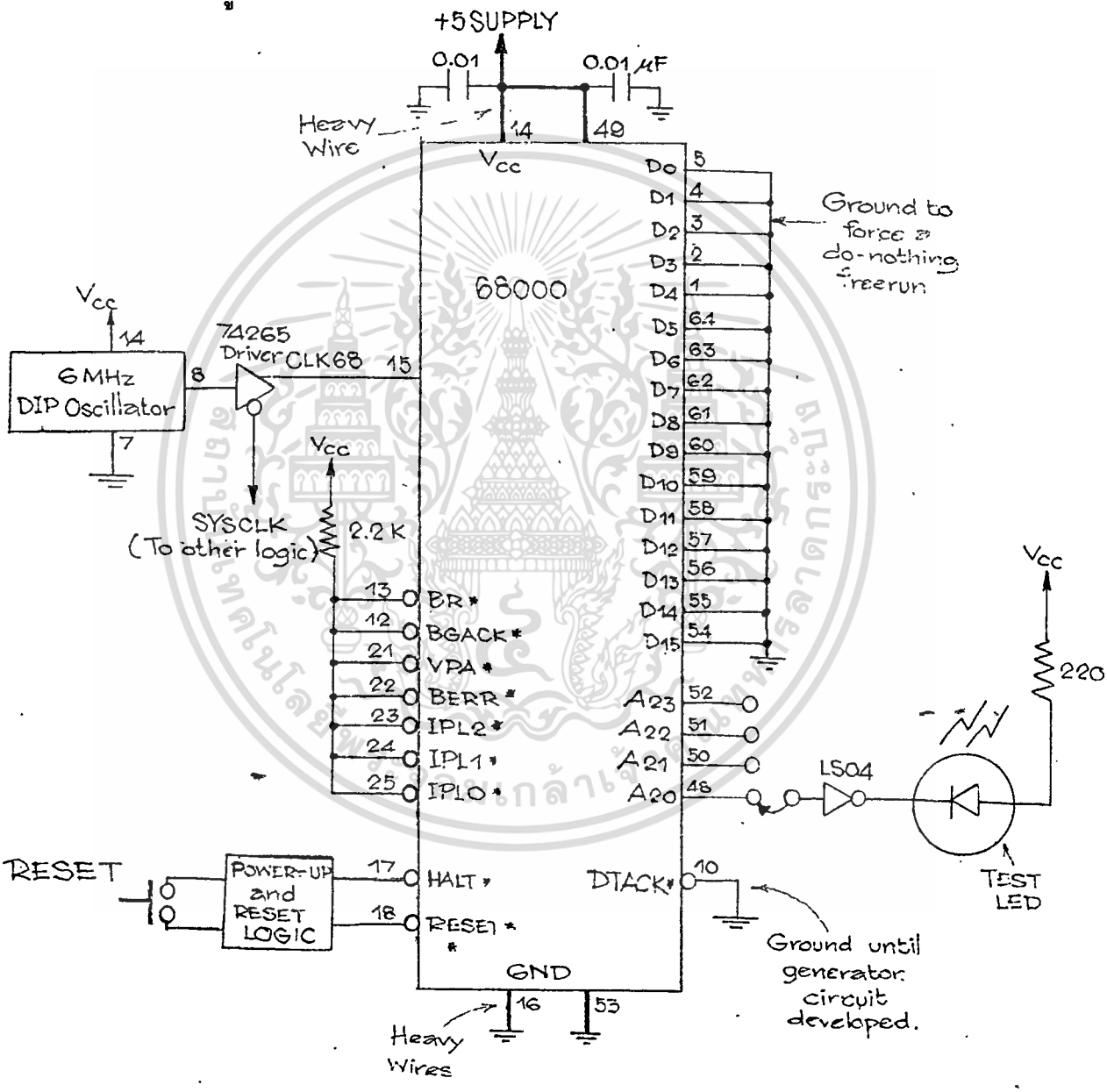
One-Word

NIL Instruction

รูปที่ 3.1 การใช้คำสั่ง NIL เพื่อทำการฟรีรัน

วิธีการนี้เรียกว่าวิธีฟรีรัน (Free Run) ซึ่งจะมีประโยชน์มากในการตรวจสอบระบบ

โดยสามารถตรวจสอบการทำงานของสัญญาณต่างๆ เพราะถือว่าขณะนี้ระบบกำลังทำงานอยู่จริง การตรวจสอบสัญญาณทำได้โดยใช้ออสซิโลสโคป (Osciloscope) หรือลอจิกโพรบ (Logic Probe) ขาที่ตรวจสอบได้ก็คือขาแอดเดรสและขาควบคุมต่างๆ ส่วนขาที่ยังไม่สามารถตรวจสอบได้ก็คือขาดาต้านั่นเอง เพราะเราได้ตัดขาดาต้าออกจากระบบแล้ว การทดสอบฟรีรันของระบบจะแสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 แสดงการทดสอบฟรีรันขององระบบ

จากภาพจะเห็นว่าขาดาต้าทั้งหมดจะถูกต่อลงกราวนด์ (Ground) ชั่วคราว เมื่อระ

บบเริ่มทำงาน SSP จะถูกเซ็ทเป็น "0" PC ถูกเซ็ทเป็น "0" ซึ่งเริ่มเฟ็ทช์ (Fetch) คำสั่งที่มีออพโค้ด 0 ระบบที่ตรวจสอบได้ขณะนี้ก็คือระบบจ่ายไฟ ระบบรีเซ็ท และระบบนาฬิกา ซึ่งถ้าไม่พบปัญหาอะไรก็แสดงว่า MC 68000 ทำงานสัมพันธ์กับสัญญาณนาฬิกา และในความเป็นจริงออพโค้ด 0 ก็ไม่ใช่คำสั่ง NOP คำสั่ง NOP จริงๆแล้วจะมีออพโค้ดเป็น 4E71H แต่ถ้าต่อให้ขาดาต้าเป็น 4E71 แล้วระบบฟรีรันจะไม่ทำงาน ดังนั้นเราจึงเรียกออพโค้ด 00 เป็นคำสั่ง NIL ซึ่งการทำงานคล้ายๆกับคำสั่ง NOP ถ้าค่าแอดเดรสที่โปรแกรมเคาน์เตอร์ไม่เป็นเลขคู่ MC 68000 จะพบว่าแอดเดรสผิดและจะเริ่มเข้าสู่ขบวนการเอ็กซ์เซพท์ชั่นทันที โดยหยุดการทำขบวนการทั้งหมด และมีสัญญาณ HALT ออกมาจาก MC 68000

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

ประการแรก เพราะเป็นชุดคำสั่งที่เป็นเลขคู่

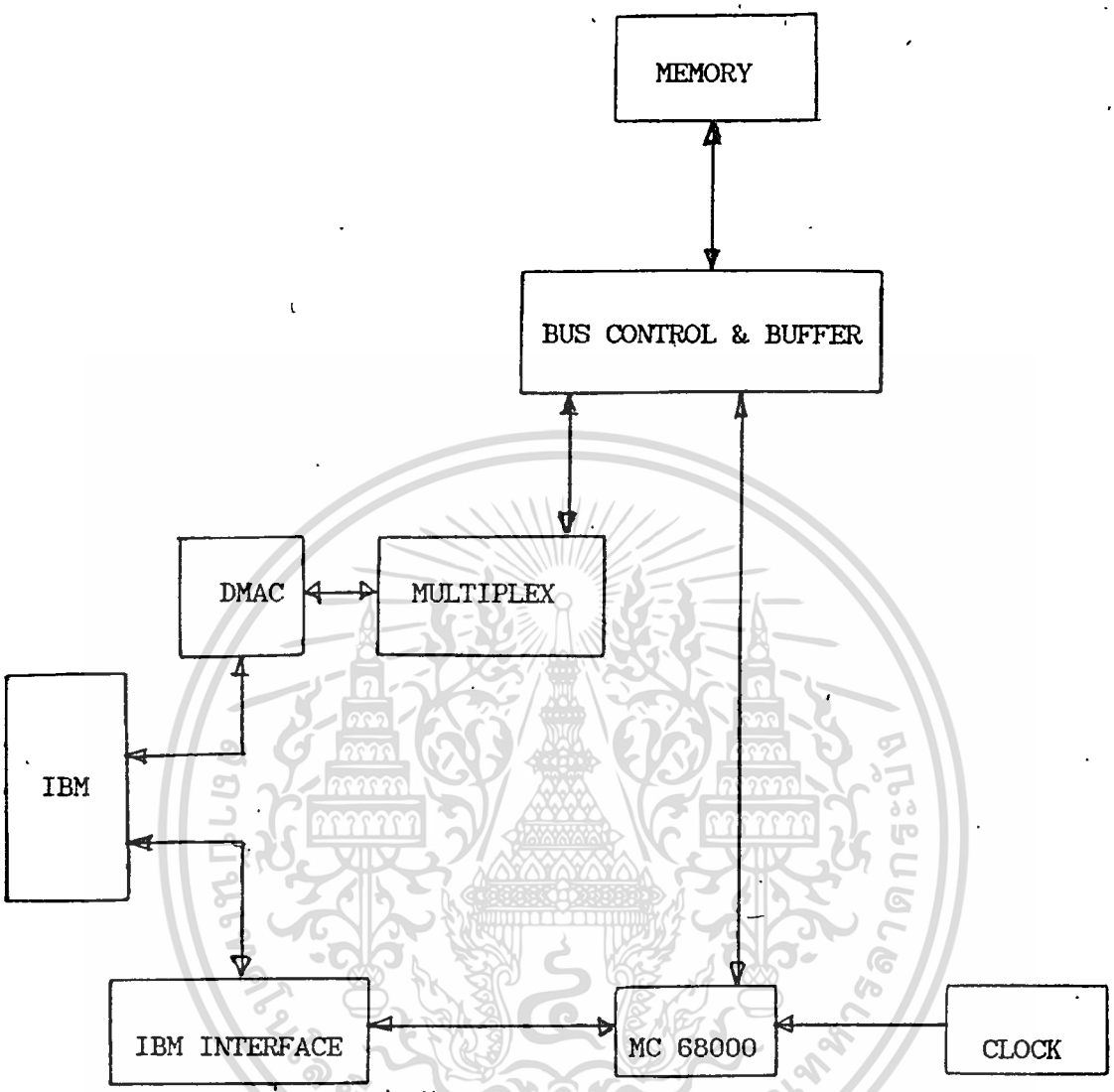
และประการที่สอง เพราะเป็นการแน่นอนกว่าที่จะต่อขาดาต้าลงกราวนด์ทั้งหมด ดีกว่าที่จะให้ขาใดขาหนึ่งเป็น"1"

หน่วยความจำซึ่งมองเห็นได้จากซีพียู 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

3.2 การออกแบบระบบ

ระบบที่ใช้ในโครงงานนี้มีรายละเอียดแยกเป็นบล็อค (Block) ได้ดังรูปที่ 3.3

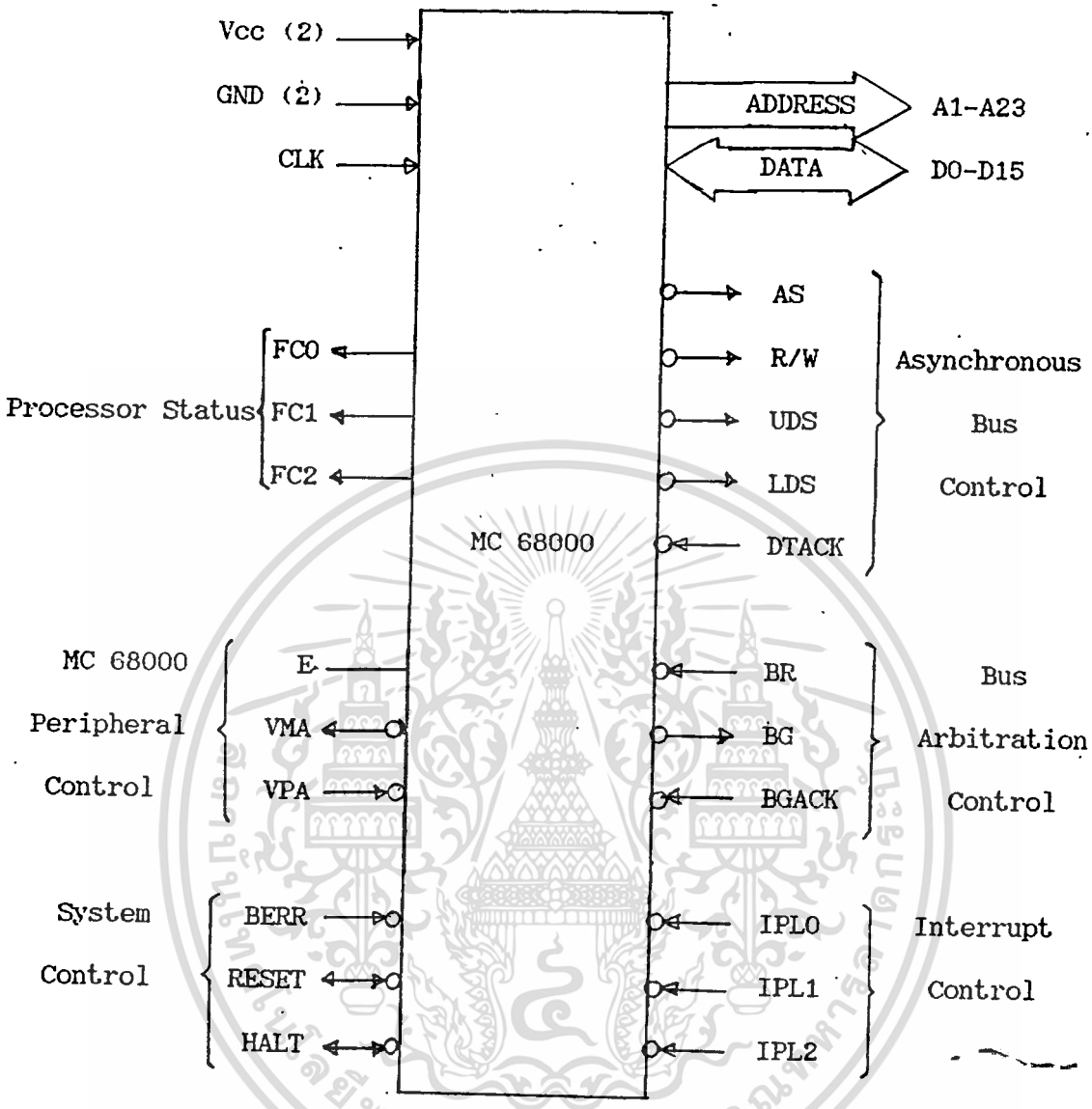


รูปที่ 3.3 บล็อคไดอะแกรมแสดงส่วนประกอบของระบบ

เพื่อเป็นการง่ายต่อการพิจารณา เราจะมาพิจารณารายละเอียดของระบบแยกเป็นบล็อคๆดังนี้

1. ส่วน MC 68000

MC 68000 มีสัญญานอินพุทและเอาท์พุทซึ่งแบ่งตามหน้าที่ดังแสดงในภาพ 3.4



รูปที่ 3.4 โครงสร้างของ MC 68000

1.1 ฟังก์ชั่นโค้ดเอาท์พุท (FC0, FC1, FC2) จะเป็นตัวบอกสถานะการทำงานของ MC 68000 ว่าจะทำงานโหมดใดดังตาราง

FC2	FC1	FC0	Cycle Type	FC2	FC1	FC0	Cycle Type
Low	Low	Low		High	Low	Low	
Low	Low	High	User Data	High	Low	High	Supervisor Data
Low	High	Low	User Program	High	High	Low	Supervisor Program
Low	High	High		High	High	High	Interrupt Acknowledge

1.2 สัญญานควบคุมระบบ (System Control) ใช้สำหรับการรีเซ็ท การฮอลท์ หรือเมื่อมีการผิดพลาดของบัสเกิดขึ้น

1.3 สัญญานควบคุมการอินเทอร์รัพท์ (Interupt Control) ระดับไพรออริตี้อินเทอร์รัพท์ทั้ง 3 ตัว (IP0, IP1, IP2) จะมีค่าสูงสุดเท่ากับ 7 MC 68000 จะทำการอินเทอร์รัพท์ได้ สัญญาณอินเทอร์รัพท์ที่เข้ามาจะต้องมีระดับไพรออริตี้สูงกว่าหรือเท่ากับที่ตั้งไว้ในรีจิสเตอร์สถานะ

สำหรับรายละเอียดต่างๆของ MC 68000 ได้กล่าวแล้วในบทที่ 2

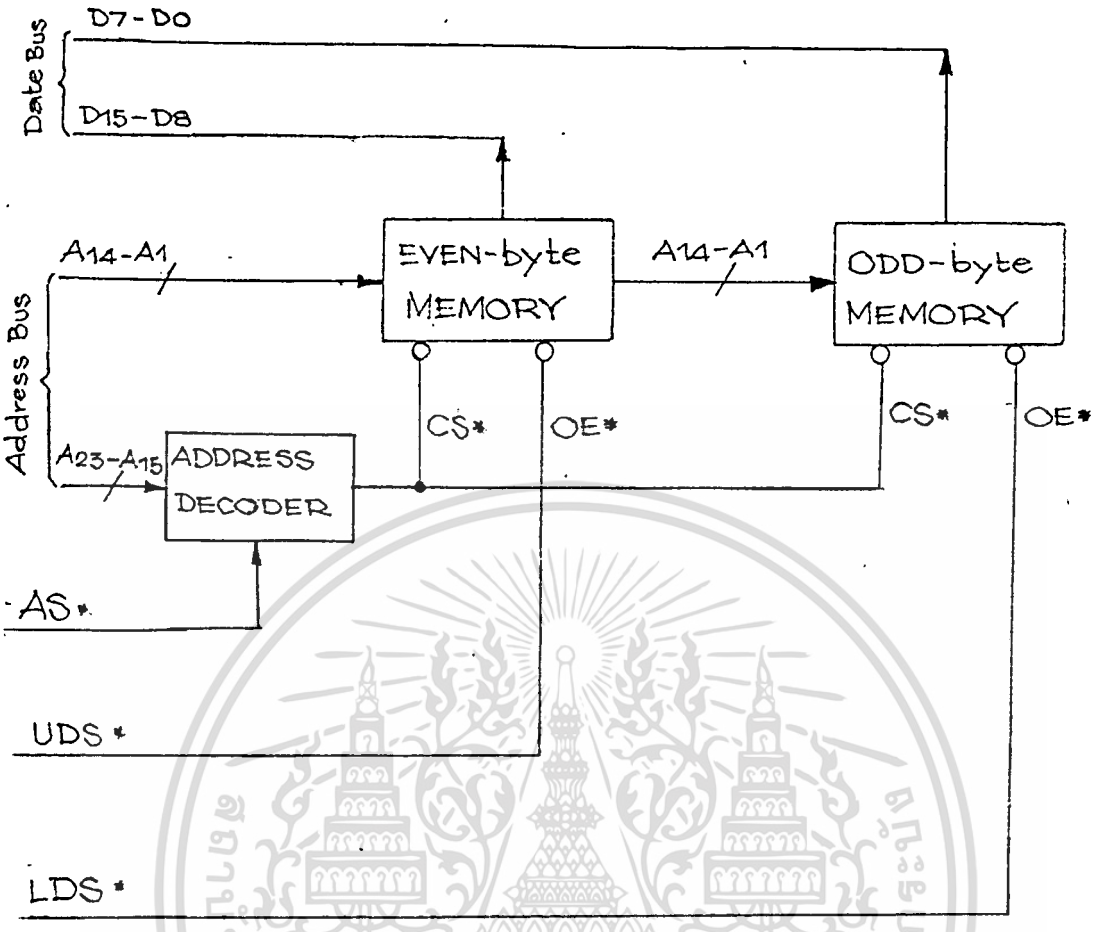
2. ส่วนสัญญาณนาฬิกา (Clock)

MC 68000 มีความสามารถที่จะรันที่ความเร็วไม่เกิน 2,4,6 หรือ 8 เมกกะเฮิร์ตตามสเป็ค (Specification) ของแต่ละตัวสำหรับเลือกใช้งาน ซึ่งความถี่เหล่านี้จะได้มาจากการนำตัวคริสตัล (Crystal) มาต่อกันเป็นวงจรออสซิเลเตอร์ หรือสามารถเลือกใช้งานจากตัวออสซิเลเตอร์ได้เลย ข้อควรระวังอย่างหนึ่งคือ ตัวออสซิเลเตอร์ไม่สามารถมีกำลังจ่ายกระแสได้สูงมากนัก ดังนั้นจึงต้องมีตัวขับกระแส (Driver) จำพวกที่ทีแอลเบอร์ 7400, 7404 หรือ 74265 แต่ก็ไม่สามารถต่อตรงจากออสซิเลเตอร์ได้เช่นกัน จึงต้องผ่านฟลิปฟลอป (Flipflop) ก่อน เนื่องจากความถี่ของออสซิเลเตอร์ส่วนมากมีค่าสูงจึงต้องสร้างวงจรเพื่อหารความถี่ให้ลดลง ในโครงงานนี้เราใช้ออสซิเลเตอร์ความถี่ 30 เมกกะเฮิร์ตผ่านฟลิปฟลอบ 2 ตัว ซึ่งฟลิปฟลอบแต่ละตัวจะให้ความถี่ลดลงครึ่งหนึ่ง ดังนั้นความถี่ออสซิเลเตอร์ขนาด 30 เมกกะเฮิร์ตจะถูกหาร 4 ได้ความถี่ที่จะนำไปใช้งานเท่ากับ 7.5 เมกกะเฮิร์ต

3. ส่วนควบคุมบัสและบัฟเฟอร์ (Bus Control and Buffer)

ส่วนความควบคุมบัสในที่นี้ก็คือ สัญญาณต่างๆซึ่งใช้เป็นอะซิงโครนัสบัสคอนโทรล สัญญาณแอดเดรสจะถูกแสดงบนแอดเดรสบัส (A1-A23) โดยที่ไม่มี A0 เพราะว่าถ้าเราให้ A0 มีค่าเป็น 0 เราจะได้แอดเดรสเป็นเลขคู่เสมอ โดยทั่วไป MC 68000 จะทำการอ่านข้อมูลที่แอดเดรสเลขคู่เสมอเมื่อมีการเฟ็ทช์คำสั่ง ดังนั้น A0 จึงไม่จำเป็นสำหรับการกระทำโปรแกรม (Program Execution) ในการรับส่งข้อมูล 8 บิทบนจะใช้สัญญาณควบคุม UDS และการรับส่งข้อมูล 8 บิทล่างจะใช้สัญญาณ LDS หรือจะใช้ทั้ง 2 สัญญานเพื่อรับส่งข้อมูลพร้อมๆกันทั้ง 16 บิท

เราสามารถใช้สัญญาณ LDS และ UDS ในการออกแบบหน่วยความจำได้ดังภาพ 3.5

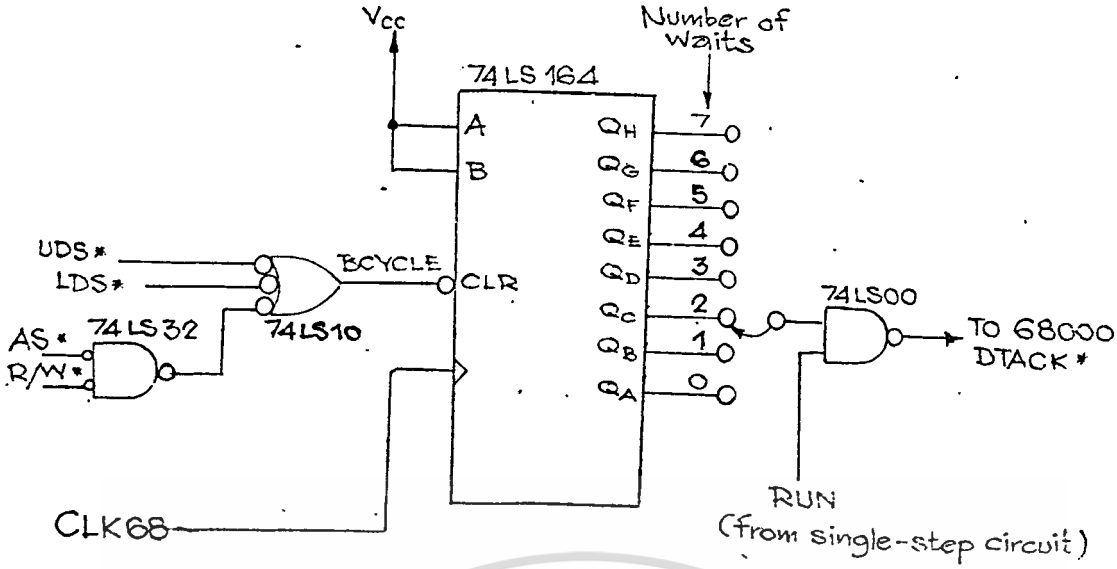


รูปที่ 3.5 การออกแบบหน่วยความจำโดยใช้สัญญาณ UDS และ LDS

สำหรับสัญญาณ DTACK จะใช้แสดงให้ MC 68000 ทราบว่าอุปกรณ์ภายนอกได้มีการทำการอ่านหรือเขียนข้อมูลเสร็จเรียบร้อยแล้ว ถ้าไม่มีสัญญาณควบคุม DTACK กลับมา 68000 จะรอไปเรื่อยๆ

ปกติสัญญาณ DTACK จะเป็น low ตั้งแต่คล็อคที่ S4 ในแต่ละไซเคิล(Cycle) และจะกลับสู่สถานะเดิมหลังจาก AS เป็น high ถ้า DTACK ไม่เป็น low ที่ S4 แล้ว MC 68000 จะสร้างสถานะ WAIT ขึ้น ในกรณีที่ต่อ DTACK เป็น low ตลอด(ลงกราวนด์) จะทำให้โปรเซเซอร์ทำงานโดยไม่มีสถานะ WAIT วงจร DTACK และ WAIT เจนเนอร์เรเตอร์แสดงดังรูปที่ 3.6

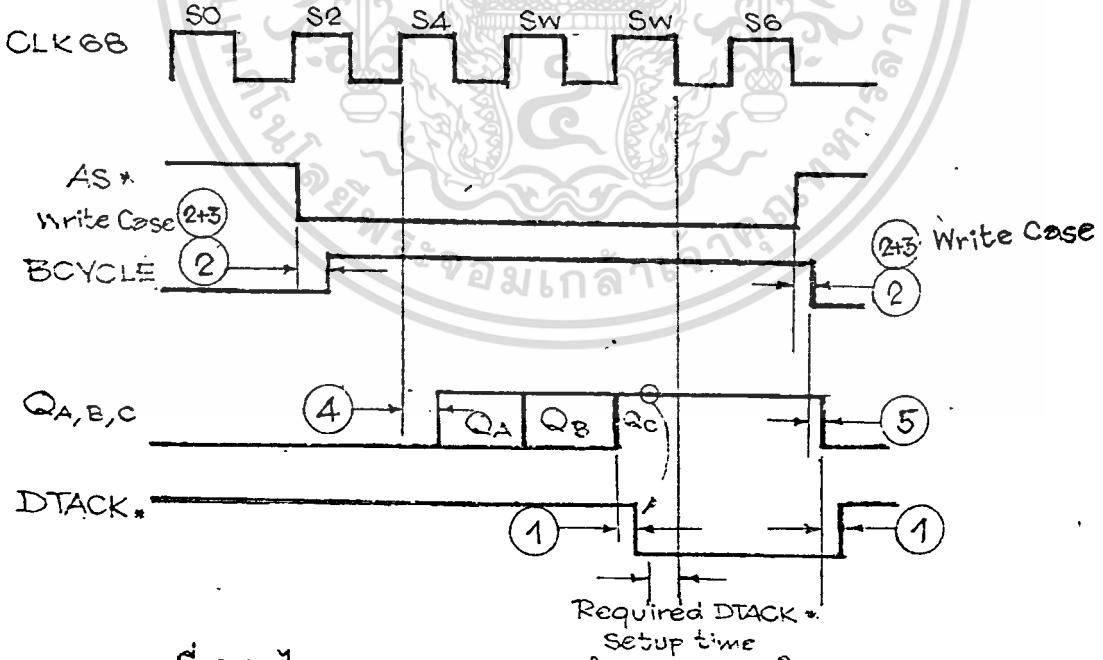
จากรูปจะเห็นได้ว่า ถ้าในสภาวะปกติ DTACK จะเป็น high แต่หลังจากนั้นบัสไซเคิลจะเริ่มทำงานไประยะหนึ่งแล้ว DTACK จะถูกทำให้เป็น low จนกระทั่งหมดไซเคิลดังรูปที่ 3.7



Propagation Delays: 74 LS 00
74 LS 10
74 LS 32
74 LS 164

t1 = 9-15 ns Average
t2 = 9-15 ns Average
t3 = 14-22 ns Average
t4 = 17-27 ns Q low-to-high from Clock.
t5 = 24-36 ns Q high-to-low from Clear

รูปที่ 3.6 วงจร DTACK และ WAIT เจนเนอร์เรเตอร์



รูปที่ 3.7 ไดอะแกรมแสดงเวลาการทำงานของวงจรในรูป 3.6

ในการรับส่งข้อมูลระหว่างหน่วยความจำกับ MC 68000 หรือหน่วยความจำกับไอบีเอ็มนั้น จะต้องผ่านบัฟเฟอร์(Buffer) เพื่อทำการพักข้อมูลและบัฟเฟอร์ยังช่วยป้องกันการผิดพลาดที่จะเกิดขึ้นอีกด้วย

4.ส่วนหน่วยความจำ(Memory)

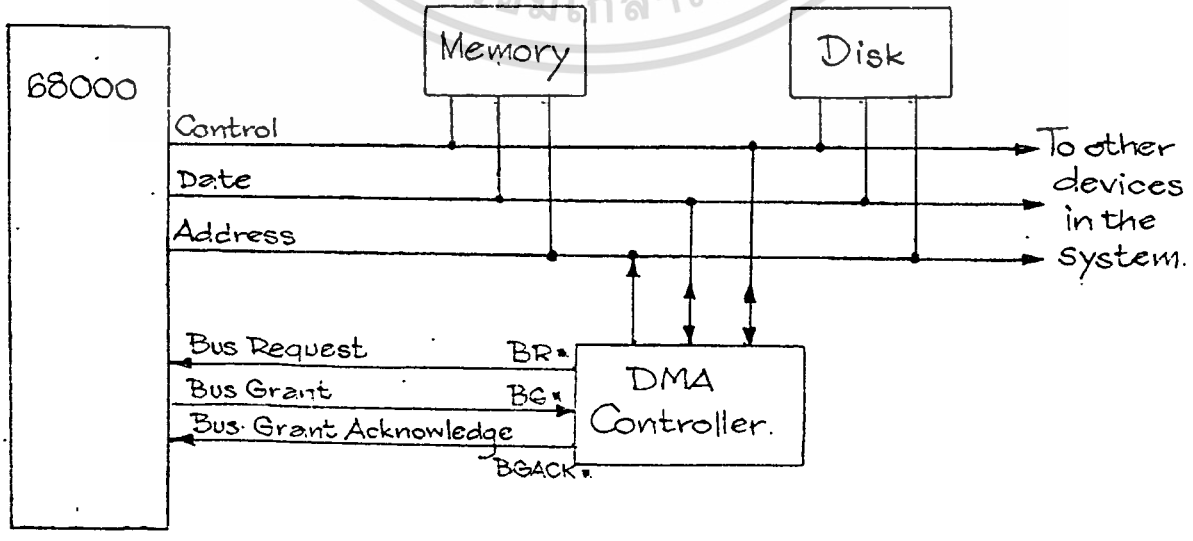
MC 68000 เป็นไมโครโปรเซสเซอร์ขนาด 16 บิท หน่วยความจำที่ใช้ในโครงงานนี้จึงใช้สแตติคแรม(Static RAM) ขนาด 32K 8 บิท จำนวน 2 ตัวมาทำเป็นหน่วยความจำของ MC 68000 การที่เราใช้สแตติคแรมเป็นหน่วยความจำ ก็เพราะว่าสามารถสร้างวงจรได้ง่าย ลดความยุ่งยากของวงจรลง แต่ข้อเสียก็คือว่ามีความจุของข้อมูลน้อย ในโครงงานนี้ เราใช้สแตติคแรม 2 ตัว ซึ่งข้อมูลขนาด 16 บิทของ MC 68000 จะถูกแบ่งไปเก็บในแรมทั้ง 2 ตัว โดยแรมตัวหนึ่งจะเก็บข้อมูลไบท์ต่ำ และอีกตัวจะเก็บข้อมูลไบท์สูง

5.ส่วนมัลติเพล็กเซอร์(Multiplexer)

เนื่องจากว่าเครื่องไอบีเอ็มมีดาต้าบัสขนาด 8 บิท แต่ MC 68000 มีดาต้าบัสขนาด 16 บิท ดังนั้น เพื่อให้เกิดการติดต่อระหว่างเครื่องไอบีเอ็มกับหน่วยความจำของ MC 68000 เกิดขึ้นได้ เราจึงต้องมีส่วนมัลติเพล็กเซอร์ให้ทำการถ่ายเทข้อมูลผ่านดาต้าบัสขนาด 8 บิทไป 16 บิท หรือ 16 บิทมา 8 บิทได้

6.ส่วนควบคุมการเข้าถึงหน่วยความจำโดยตรง(Direct Memory Access Control:DMAC)

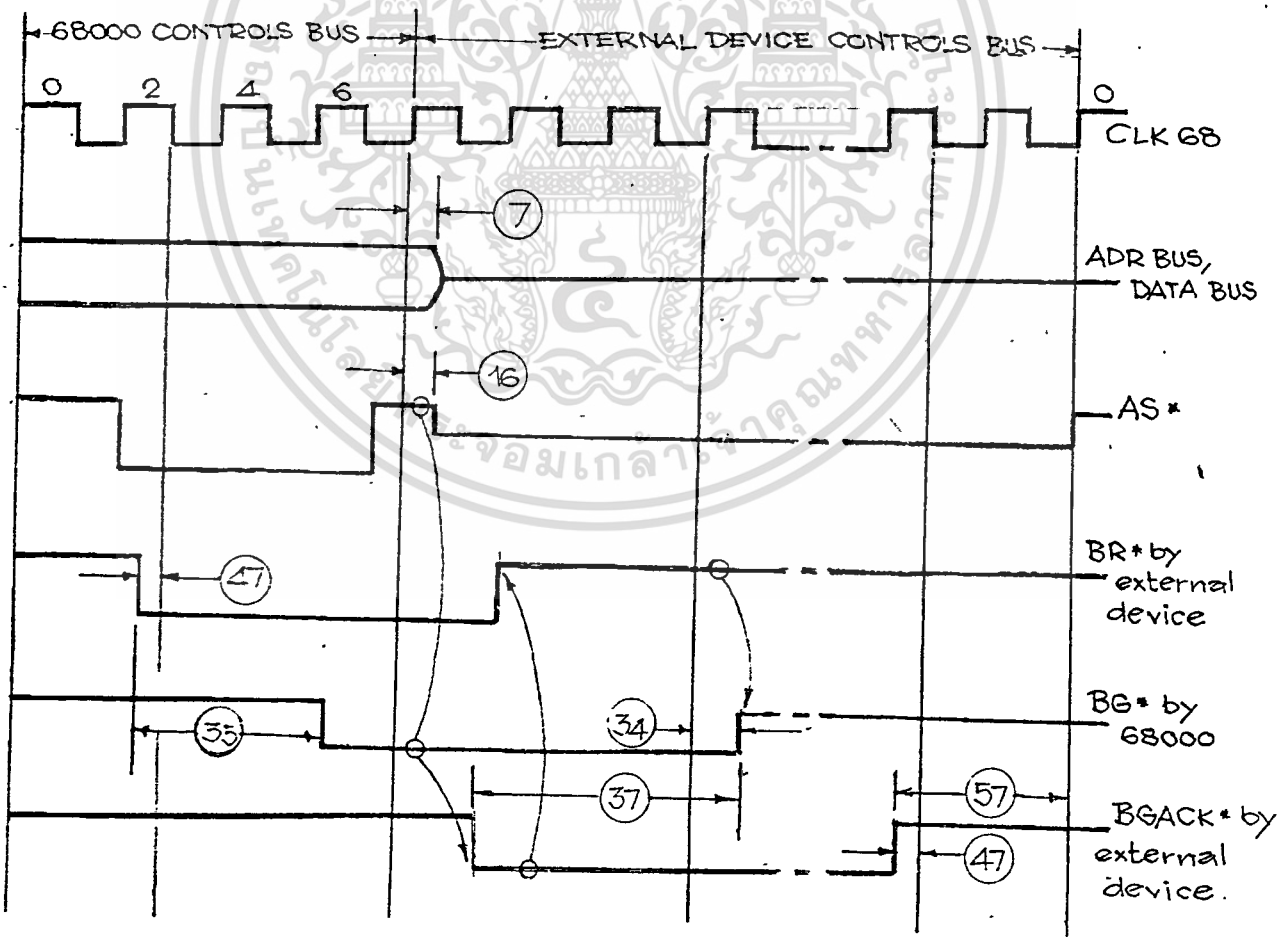
ส่วนนี้คือส่วนที่เป็นการควบคุมบัสอาร์บิเทรชั่น(Bus Arbitration Control) โดยมีสัญญาณ BR ซึ่งเป็นสัญญาณอินพุท สัญญาณ BG ซึ่งเป็นสัญญาณเอาท์พุท และสัญญาณ BGACK ซึ่งเป็นสัญญาณอินพุท สัญญาณทั้งสามนี้จะเป็นสัญญาณควบคุมบัสสำหรับการทำ DMA การทำ DMA ก็คือการแลกเปลี่ยนข้อมูลโดยตรงระหว่างไอบีเอ็มกับหน่วยความจำโดยไม่ผ่านตัว MC 68000



รูปที่ 3.8 บล็อกไดอะแกรมแสดงการควบคุมการทำดีเอ็มเอ(DMA)

จากภาพที่ 3.8 จะเห็นว่าตัวควบคุมการทำ DMA (DMAC) จะเชื่อมต่อกับบัสของ 68000 และเมื่อต้องการติดต่อข้อมูลระหว่างหน่วยความจำกับดิสค์ DMAC จะทำการส่งสัญญาณ BR ไปยัง 68000 เพื่อบอกให้ทราบว่าต้องการทำการ DMA 68000 ก็จะตอบรับทราบกลับมาโดยสัญญาณ BG DMAC ก็จะรอจนกว่า 68000 จะเสร็จสิ้นการทำบัสไซเคิลเสียก่อน (สัญญาณ AS จะเป็น"high") จึงให้สัญญาณ BGACK ออกไปยัง 68000 เพื่อแสดงการรับรู้ถึงสัญญาณ BG ที่ 68000 ส่งมา (68000 จะยกเลิกการใช้บัสทั้งหมดให้มีเพียง DMAC เท่านั้นที่ใช้บัสทั้งหมดนี้) และต่อมา DMAC จะทำการยกเลิก BR จึงทำให้ 68000 ยกเลิกสัญญาณ BG ขณะนี้จึงทำการติดต่อข้อมูลโดยการทำ DMA จนกระทั่ง DMAC ยกเลิกสัญญาณ BGACK 68000 ก็จะกลับมาใช้บัสทั้งหมดในการทำบัสไซเคิลต่อไป

การทำงานของภาพที่ 3.8 สามารถเขียนเป็นไดอะแกรมเวลา (Timing Diagram) ได้ดังภาพที่ 3.9



รูปที่ 3.9 ไดอะแกรมแสดงเวลาการทำงานของวงจรในรูป 3.8

7.ส่วนเชื่อมต่อกับเครื่องไอบีเอ็ม(IBM Interface)

ส่วนนี้จะทำหน้าที่ต่างๆกันตามพอร์ท(Port) ดังนี้

พอร์ท 3A8 -พอร์ทนี้ จะทำหน้าที่ ควบคุมระบบโดยมีสัญญาณควบคุมการรีเซ็ทการฮอลท์ และสัญญาณควบคุมการผิดพลาดของบัส

พอร์ท 3A9 -จะเป็นพอร์ทข้อมูล 1

พอร์ท 3AA -จะเป็นพอร์ทข้อมูล 2

พอร์ท 3AB -จะใช้ในการทำซิงเกิลสเต็ป (Single Step)ในการใช้ดีบัก(Debug)

พอร์ท 3AC -จะใช้ในการทำ DMA

พอร์ท 3AD, พอร์ท 3AE, พอร์ท 3AF } จะเป็นการทำ DMA ของแอดเดรส

<u>ผลการทำงานของระบบที่สร้าง</u>

เมื่อเราต่อวงจรส่วนต่างๆแต่ละส่วนลงบนการ์ดแล้ว เราได้นำไปทดสอบการทำงาน โดยเราจะทำการทดสอบเป็นส่วนๆดังนี้

1. ส่วนไมโครโปรเซสเซอร์และวงจรสร้างสัญญาณนาฬิกา

ในการทดสอบไมโครโปรเซสเซอร์ เราจะทำการทดสอบฟรีรัน (ได้กล่าวแล้วในหัวข้อ 3.1 ภาพที่ 3.2) จากภาพ 3.2 เมื่อเราทำการ "ON" ส่วนจ่ายกำลัง (Power-up) จะเห็นว่าที่ LED จะมีไฟสว่างและดับเกิดขึ้น ในการทดสอบฟรีรันนี้จะทำให้ทราบว่า MC 68000 สามารถทำงานสัมพันธ์กับส่วนวงจรสร้างสัญญาณนาฬิกา

2. ส่วนเชื่อมต่อกับไอบีเอ็ม

เราจะทดสอบพอร์ทต่างๆว่าสามารถทำงานได้หรือไม่ เมื่อเราทดสอบการทำงานของแต่ละพอร์ทแล้ว และแต่ละพอร์ทซึ่งมีหน้าที่ต่างๆกัน สามารถทำงานได้โดยไม่มีข้อผิดพลาด สแดงว่า MC 68000 สามารถใช้งานร่วมกับเครื่องไอบีเอ็มได้

3. ส่วนควบคุมการทำ DMA

ในการจะทดสอบการทำงานของส่วนนี้ เราจะต้องทำการเขียนโปรแกรมบนไอบีเอ็ม เพื่อทำการควบคุมการทำ DMA ระหว่างหน่วยความจำของการ์ดกับหน่วยความจำของไอบีเอ็ม โดยเราจะเขียนเป็นโปรแกรมเรซิเด็นท์ (Resident) โปรแกรมเรซิเด็นท์ก็คือ โปรแกรมที่โหลดไปรันบนไอบีเอ็มแล้วฝังตัวเป็นส่วนหนึ่งของดอส (DOS) ส่วนใหญ่จะเป็นลักษณะโปรแกรม

ที่ใช้เป็นอินเทอร์รัพท์เซอร์วิสรูทีน (Interupt Service Routine) ใช้ในการแก้อินเทอร์รัพท์ให้จัดการกับการทำ DMA

เมื่อเขียนโปรแกรมดังกล่าวเสร็จหมดแล้ว เราสามารถทดสอบการทำ DMA โดยให้มีการติดต่อข้อมูลระหว่างหน่วยความจำของไอบีเอ็มกับหน่วยความจำของการ์ด โดยดูว่าค่าข้อมูลที่ถูกติดต่อโดยการทำ DMA มีการเปลี่ยนแปลงไปหรือไม่ ถ้ามีการเปลี่ยนแปลงแสดงว่าส่วน DMAC ในโครงงานนี้มีปัญหา เราจะต้องทำการตรวจสอบส่วนนี้ รวมถึงการตรวจสองส่วนมัลติเพล็กซ์เซอร์, ส่วนบัพเฟอร์ และส่วนการควบคุมบัสด้วย

บทที่ 4

การออกแบบส่วนซอฟต์แวร์

4.1 บทนำ

ในบทนี้เราจะกล่าวถึงการออกแบบโปรแกรมแอสเซมเบลอร์, ดีบักเกอร์ และโหลดเดอร์ ว่ามีขั้นตอนในการออกแบบและพัฒนาอย่างไร

แอสเซมเบลอร์ เป็นโปรแกรมที่ทำหน้าที่แปลจากภาษาแอสเซมบลี้ไปเป็นภาษาเครื่องซึ่งในที่นี้ก็คือภาษาเครื่องของ 68000 การที่ต้องพัฒนาโปรแกรมนี้ขึ้นมาก็เนื่องจากการเขียนโปรแกรมด้วยภาษาเครื่องของ 68000 นั้นเป็นเรื่องที่เสียเวลา ยากต่อการเขียนและยังมีโอกาสผิดพลาดได้มากอีกด้วย อย่างไรก็ตามก็ยังมีโปรแกรมประเภทนี้อยู่บ้าง แต่จะมีราคาที่ค่อนข้างแพงจึงได้ตัดสินใจพัฒนาขึ้นมาใช้เอง ทั้งยังทำให้การพัฒนาต่อในอนาคตทำได้ง่ายกว่าอีกด้วย

ดีบักเกอร์ ถือได้ว่าเป็นสิ่งจำเป็นที่จะต้องมีในระบบพัฒนา (Development System) เพราะเป็นสิ่งเดียวที่จะช่วยในการติดตามการทำงานของไมโครโปรเซสเซอร์ ซึ่งในการพัฒนาระบบต่างๆขึ้นมานั้นความผิดพลาดย่อมมีโอกาสเกิดขึ้นได้เสมอ ตัวดีบักเกอร์นี้จะเป็นเครื่องมือที่จะช่วยในการหาสาเหตุแห่งความผิดพลาดเหล่านั้นได้

โหลดเดอร์ โปรแกรมนี้จะใช้โหลดโปรแกรมที่ได้เขียนและทดสอบการทำงานแล้วว่าถูกต้องลงบนการ์ด 68000 เพื่อทดลองการทำงานจริง

4.2 การออกแบบโปรแกรมแอสเซมเบลอร์

4.2.1 โครงสร้างของโปรแกรมแอสเซมบลี้

โปรแกรมภาษาแอสเซมบลี้โดยทั่วไปจะมีส่วนประกอบ 4 ส่วนด้วยกันดังนี้

4.2.1.1 ส่วนลาเบล (Label) เป็นส่วนที่เป็นเครื่องหมาย (Symbolic) ที่จะใช้ในการกระโดด (Jump) โดยลาเบลจะต้องขึ้นต้นด้วยตัวอักษรและจะตามด้วยตัวอักษรหรือตัวเลขหรือขีดล่างเท่านั้น

4.2.1.2 ส่วนมีโมนิค (Mnemonic) เป็นส่วนที่เป็นคำสั่งภาษาแอสเซมบลี้ โดยจะมีหน้าที่บอกการกระทำของคำสั่งนั้น

4.2.1.3 ส่วนโอเปอร์แรนด์ (Operand) จะบอกถึงตัวกระทำและตัวที่ถูกกระทำ โดยที่ตัวกระทำและตัวถูกกระทำนี้อาจเป็นรีจิสเตอร์หรือหน่วยความจำก็ได้

4.2.1.4 ส่วนคอมเมนต์ (Comment) เป็นส่วนที่ไม่มีผลต่อการทำงานของโปรแกรมแต่เป็นส่วนที่แอสเซมเบลอร์อนุญาติให้มีเพื่อใช้อธิบายการทำงานของโปรแกรม ทั้งนี้เพราะภาษาแอสเซมบลี้เป็นภาษาที่ยากต่อการเข้าใจดังนั้นส่วนนี้จึงนับได้ว่าเป็นส่วนที่จำเป็น และควรจะเขียนไว้ในโปรแกรม แต่ในการแปลแล้วตัวแอสเซมเบลอร์จะไม่สนใจส่วนดังกล่าวนี้เลย

4.3 โครงสร้างของโปรแกรมแอสเซมเบลอร์

สำหรับหน้าที่ของโปรแกรมแอสเซมเบลอร์นั้น โดยหลักการแล้วก็คือโปรแกรมที่นำ ซอร์สโปรแกรม (Source Program : หมายถึงโปรแกรมภาษาแอสเซมบลี้ที่เราต้องการแปลให้เป็นภาษาเครื่อง) มาแปลเป็นออบเจ็คโปรแกรม (Object Program : หมายถึงโปรแกรมภาษาเครื่องที่ผ่านการแปลมาจากในการแปลภาษาไม่ว่าจะเป็นภาษาอะไรก็ตาม มักจะประกอบด้วยขั้นตอนต่างๆ ดังต่อไปนี้

4.3.1 ส่วนวิเคราะห์คำ (Lexical Analysis)

ส่วนนี้จะทำหน้าที่ในการแยกตัวโปรแกรมออกเป็นส่วนต่างๆ เช่นถ้าเป็นตัวแปลภาษาประเภทแอสเซมเบลอร์ก็จะแยกออกเป็น ส่วนลาเบล, ส่วนนีโมนิค, ส่วนโอเปอร์แรนด์, ส่วนคอมเมนต์ จากนั้นก็จะนำส่วนต่างๆ ที่แยกได้นี้ไปใช้ในขั้นตอนอื่นๆ

4.3.2 ส่วนวิเคราะห์โครงสร้าง (Syntax Analyzer)

ส่วนนี้จะทำหน้าที่ตรวจสอบโครงสร้างของตัวโปรแกรม ว่าถูกต้องตรงตามไวยากรณ์ของภาษานั้นๆ หรือไม่ถ้าพบว่าเกิดความผิดพลาดเกิดขึ้นในโปรแกรม ก็จะรายงานความผิดพลาดเหล่านั้นแก่ผู้ใช้ เพื่อให้ผู้ใช้ได้แก้ไขตัวโปรแกรมให้ถูกต้อง

4.3.3 ส่วนสร้างรหัสภาษาเครื่อง (Code Generater)

เป็นส่วนที่นำโปรแกรมที่ได้ตรวจสอบโครงสร้างแล้วว่าถูกต้องมาทำการสร้างรหัสที่เป็นภาษาเครื่องที่มีการทำงานตรงกับคำสั่งในซอร์สโปรแกรมซึ่งผลลัพธ์ที่ได้จากส่วนนี้จะเรียกว่าออบเจ็คโปรแกรมซึ่งสามารถนำไปทำงานได้ทันที

4.4 ภาษาแอสเซมบลี้ของ MC68000

ภาษาแอสเซมบลี้ของ MC68000 นั้นนับได้ว่าเป็นภาษาแอสเซมบลี้ที่มีโครงสร้างของภาษาดีมาก อีกทั้งมีโหมดในการอ้างแอดเดรสมาก ทำให้ง่ายต่อการเขียนโปรแกรม นอกจากนั้นชุดคำสั่งยังมีให้เลือกใช้มาก ซึ่งเราสามารถแบ่งชุดคำสั่งดังกล่าวออกเป็นกลุ่มๆ ได้ดังต่อไปนี้

4.4.1 กลุ่มคำสั่งที่ทำการเคลื่อนย้ายข้อมูล

เป็นกลุ่มของคำสั่งที่มีหน้าที่เกี่ยวกับการเคลื่อนย้ายข้อมูลไม่ว่าจะเป็น ระหว่างรีจิสเตอร์ ระหว่างหน่วยความจำหรือระหว่างรีจิสเตอร์กับหน่วยความจำซึ่งคำสั่งที่เกี่ยวกับการเคลื่อนย้ายข้อมูลนี้แสดงดังตารางที่ 4.1

Mnemonic	Operation
EXG	Exchange registers
LEA	Load effective address
LINK	Link and allocate stack
MOVE	Move source to destination
MOVEA	Move source to address register
MOVEC	Move control register
MOVEM	Move multiple registers
MOVEP	Move to peripheral
MOVEQ	Move short data to destination
MOVES	Move address space
PEA	Push effective address
UNLK	Unlink stack

ตารางที่ 4.1 แสดงคำสั่งที่ใช้ในการเคลื่อนย้ายข้อมูล

จากตารางจะเห็นได้ว่าไม่มีคำสั่งที่เกี่ยวกับสแต็คเช่น PUSH หรือ POP ทั้งนี้เพราะว่าไมโครโปรเซสเซอร์ตัวนี้สามารถทำงานในแบบที่เป็นสแต็คได้โดยใช้คำสั่ง MOVE และข้อสังเกตุอีกประการก็คือไม่มีคำสั่งที่ทำการเคลื่อนย้ายข้อมูลเป็นบล็อก แต่การเคลื่อนย้ายเป็นบล็อกก็ยังสามารถทำได้โดยใช้โหมดของการอ้างแอดเดรสรวมกับคำสั่งที่ทำงานเป็นลูป

4.4.2 คำสั่งที่มีการกระทำทางคณิตศาสตร์

ใน MC68000 ได้ให้คำสั่งที่มีการกระทำทางคณิตศาสตร์ไว้ 4 แบบด้วยกันคือ บวก ลบ คูณและหาร ดังแสดงในตารางที่ 4.2

4.4.3 คำสั่งที่มีการกระทำทางตรรก

ใน MC68000 สามารถใช้คำสั่งที่มีการกระทำทางตรรกได้ 4 แบบคือ AND OR XOR และ NOT ซึ่งผลของคำสั่งประเภทนี้จะเป็นได้ 2 ค่าคือจริงและเท็จ (True และ False) ซึ่งคำสั่งประเภทนี้แสดงได้ดังตารางที่ 4.3

4.4.4 คำสั่งประเภทชิฟท์และโรเทต

คำสั่งประเภทนี้เป็นการคำสั่งที่มีการทำงานเกี่ยวกับการเลื่อนบิตซึ่งสามารถนำคำสั่งประเภทนี้ไปประยุกต์ใช้เป็นการคูณหรือหารได้อีกด้วยแสดงได้ดังตารางที่ 4.4

Mnemonic	Operation
ADD	Add source to destination
ADDA	Add source to address register
ADDI	Add immediate data to destination
ADDQ	Add short data to destination
ADDX	Add with extend bit to destination
CLR	Clear operand
CMP	Compare source to destination
CMPA	Compare source to address register
CMPI	Compare immediate data to destination
CMPM	Compare memory
CMP2*	Compare register to upper/lower bounds
DIVS	Signed divide
DIVU	Unsigned divide
DIVSL*	Long signed divide
DIVUL*	Long unsigned divide
EXT	Sign extend
EXTB	Sign extend byte
MULS	Signed multiply
MULU	Unsigned multiply
NEG	Negate
NEGX	Negate with extend
SUB	Subtract source from destination
SUBA	Subtract source from address register
SUBI	Subtract immediate from destination
SUBQ	Subtract short from destination
SUBX	Subtract with extend bit from destination

*68020 only

ตารางที่ 4.2 แสดงคำสั่งประเภทการกระทำทางคณิตศาสตร์

Mnemonic	Operation
AND	AND source to destination
ANDI	AND immediate data to destination
EOR	Exclusive OR source to destination
EORI	Exclusive OR immediate data to destination
NOT	NOT destination
OR	OR source to destination
ORI	OR immediate data to destination
S_{cc}	Test condition codes and set operand
TST	Test operand and set condition codes

ตารางที่ 4.3 แสดงคำสั่งที่มีการทำงานทางตรรก

4.4.5 คำสั่งที่มีการทำงานกับบิต

การทำงานของคำสั่งในกลุ่มนี้จะสามารถกระทำกับบิตได้โดยตรง เช่นเซ็ต รีเซ็ต หรือการตรวจสอบบิต ซึ่งคำสั่งประเภทนี้จะมีประโยชน์ในแง่ของการใช้ข้อมูลในหน่วยความจำเพื่อทำหน้าที่เป็นแฟล็ก ซึ่งคำสั่งประเภทนี้จะช่วยให้สามารถใช้ข้อมูล 1 ตัวเป็นแฟล็กได้หลายตัว คำสั่งประเภทนี้แสดงได้ดังตารางที่ 4.5

Mnemonic	Operation
ASL	Arithmetic shift left
ASR	Arithmetic shift right
LSL	Logical shift left
LSR	Logical shift right
ROL	Rotate left
ROR	Rotate right
ROXL	Rotate left with extend bit
ROXR	Rotate right with extend bit
SWAP	Swap words of a long word

ตารางที่ 4.4 แสดงคำสั่งที่มีการทำงานแบบชิฟท์และโรเทต

Mnemonic	Operation
BCHG	Change bit
BCLR	Clear bit
BSET	Set bit
BTST	Test bit

ตารางที่ 4.5 แสดงคำสั่งที่มีการทำงานแบบบิต

4.4.6 คำสั่งที่มีการทำงานกับเลขบีซีดี (BCD : Binary Code Decimal)

ในการทำงานบางอย่างเราจะเป็นจะต้องใช้ตัวเลขแบบบีซีดีซึ่งใน MC68000 ได้ทำคำสั่งสนับสนุนการทำงานประเภทนี้ไว้ 2 แบบคือบวกและลบเท่านั้น คำสั่งในกลุ่มนี้แสดงดังตารางที่ 4.6

Mnemonic	Operation
ABCD	Add source to destination
NBCD	Negate destination
PACK*	Pack source to destination
SBCD	Subtract source from destination
UNPK*	Unpack source to destination
*68020 only	

ตารางที่ 4.6 แสดงคำสั่งที่มีการทำงานแบบบีซีดี

4.4.7 คำสั่งประเภทควบคุมการทำงานของโปรแกรม

ในกลุ่มคำสั่งนี้นับได้ว่าเป็นกลุ่มคำสั่งที่จะขาดเสียไม่ได้ ทั้งนี้เพราะในการเขียนโปรแกรมเพื่อให้สามารถทำงานตามที่เราต้องการนั้น โปรแกรมจะต้องทำงานตามเงื่อนไขต่างๆ ที่ได้วางไว้ซึ่งกลุ่มคำสั่งนี้เองที่จะใช้ควบคุมการทำงานของโปรแกรมให้เป็นไปตามเงื่อนไข คำสั่งในกลุ่มนี้แสดงดังตารางที่ 4.7

Mnemonic	Operation
B_{cc}	Branch conditionally
BRA	Branch unconditionally
BSR	Branch to subroutine
CALLM*	Call module
DB_{cc}	Test, decrement, and branch
JMP	Jump to address
JSR	Jump to subroutine
NOP	No operation
RTD**	Return and deallocate stack
RTE+	Return from exception
RTM*	Return from module
RTR	Return and restore condition codes
RTS	Return from subroutine
+privileged instruction	
*68020 only	
**68010-68020 only	

ตารางที่ 4.7 แสดงกลุ่มคำสั่งที่ควบคุมการทำงานของโปรแกรม

4.4.8 กลุ่มคำสั่งที่ควบคุมการทำงานของระบบ

กลุ่มคำสั่งนี้เป็นกลุ่มคำสั่งพิเศษที่ไม่ได้มีการใช้งานบ่อยนัก มักจะใช้ในการควบคุมการทำงานของตัวซีพียู เช่นการเปลี่ยนสถานะการทำงานจากสถานะผู้ใช้ (User State) ไปเป็นสถานะซูเปอร์ไวเซอร์ (Supervisor State) เป็นต้น คำสั่งในกลุ่มนี้แสดงได้ดังตารางที่ 4.8

Mnemonic	Operation
ANDI	AND immediate to status register/condition code register
BKPT	Breakpoint trap
CHK	Trap on upper out-of-bounds operand
CHK2*	Trap on out-of-bounds operand
EORI	Exclusive OR immediate to status
ILLEGAL	Illegal instruction trap
MOVE	Move to/from status register/condition code register
MOVEC+	Move to/from control register
MOVES+	Move to/from address space
RESET+	Assert RESET line
STOP+	Stop processor
TRAP	Trap unconditionally
$TRAP_{cc}$*	Trap on condition
TRAPV	Trap on overflow

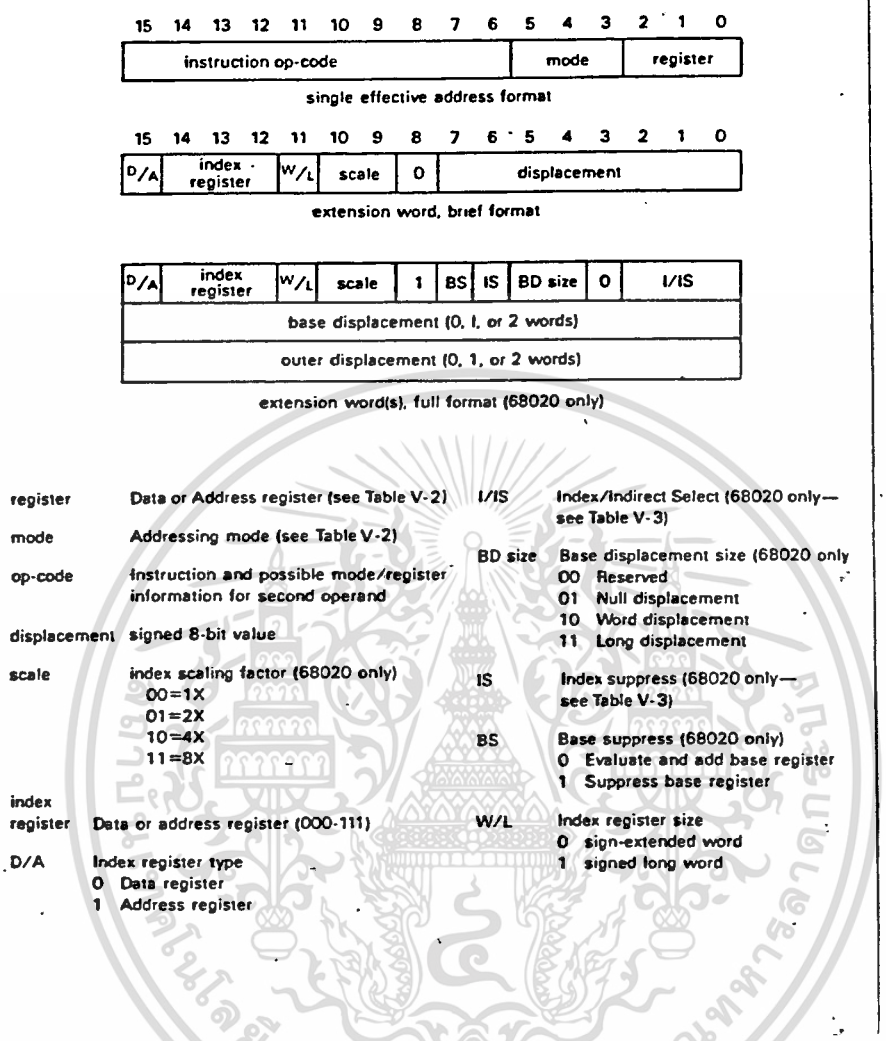
+privileged instruction
*68020 only

ตารางที่ 4.8 แสดงกลุ่มคำสั่งที่ควบคุมการทำงานของระบบ

4.5 ข้อสังเกตจากคำสั่งของ MC68000

โครงสร้างของภาษาแอสเซมบลี้ MC68000 นั้นค่อนข้างจะเป็นรูปแบบที่ตายตัว กล่าวคือในแต่ละคำสั่ง บิตแต่ละบิตจะมีความหมายใกล้เคียงกัน เช่น ในคำสั่งประเภทเดียวกัน (เช่น ADDI SUBI) จะสามารถแยกว่าเป็นคำสั่งใดได้จากบิตที่ 12-15 บิตที่ 0-5 ถ้าเป็นคำสั่งประเภทแอฟเฟคทีฟแอดเดรส (Effective Address)

สำหรับความหมายของบิตต่างๆ จะแสดงได้ดังในรูปที่ 4.1



รูปที่ 4.1 แสดงความหมายของบิตต่างๆ ของคำสั่ง

4.6 การพัฒนาโปรแกรม

4.6.1 การพัฒนาโปรแกรมแอสเซมเบลอร์

จากข้อมูลที่แสดงความหมายของแต่ละบิตในแต่ละคำสั่ง เมื่อเรานำมาหาความแตกต่างระหว่างคำสั่ง ทำให้เราสามารถทำการแยกคำสั่งเหล่านี้ออกเป็นกลุ่มต่างๆ (กลุ่มในที่นี้ คนละความหมายกับกลุ่มการทำงานที่ได้กล่าวมาแล้ว) โดยแต่ละกลุ่มจะมีโครงสร้างของคำสั่งที่เหมือนกัน การที่เราแบ่งเป็นกลุ่มนี้ก็เพื่อที่จะสามารถเขียนเป็นโปรแกรมที่มีขนาดเล็กที่สุดเท่าที่จะทำได้ ในโปรแกรมแอสเซมเบลอร์นี้เราจะแบ่งการทำงานออกเป็น 2 ส่วนคือ

4.6.1.1 ส่วนที่ 1 เป็นส่วนที่ทำหน้าที่วิเคราะห์คำ (Lexical

Analysis) และ วิเคราะห์โครงสร้าง (Syntax Analysis) กล่าวคือการทำงานในส่วนนี้จะทำการแยกซอร์สโปรแกรมที่เป็นภาษาแอสเซมบลี้ออกเป็นส่วนต่างๆ ดังที่ได้กล่าวเอาไว้ข้างต้น จากนั้นก็นำส่วนที่เป็นลาเบลไปเก็บไว้ในตาราง เพื่อจะนำไปใช้อ้างอิงกับแอดเดรสของลาเบลนั้นอีกครั้ง นอกจากนั้นหน้าที่อีกประการของส่วนนี้ก็คือทำการตัดส่วนของโปรแกรมที่ไม่จำเป็นต่อการแปลออกไปซึ่งได้แก่คอมเม้นท์ต่างๆ โดยที่ในขณะที่ทำการแยกส่วนต่างๆ อยู่นั้นก็จะทำการตรวจสอบโปรแกรมว่าตรงตามไวยากรณ์ที่กำหนดหรือไม่ ซึ่งถ้าไม่ตรงก็จะแสดงข้อผิดพลาดออกมา ดังนั้นเมื่อผ่านการทำงานในขั้นตอนที่ 1 แล้วเราจะได้ตารางของลาเบลที่มีแอดเดรสของแต่ละลาเบลกำกับไว้ และเราจะได้โปรแกรมที่อยู่ในรูปแบบของรหัสชั่วคราว (Intermediate Code) ซึ่งเป็นรหัสที่เหมาะสมที่จะใช้งานในขั้นตอนต่อไป

4.6.1.2 ส่วนที่ 2 ในส่วนนี้จะเป็นการทำงานในขั้นตอนวิเคราะห์โครงสร้าง (Syntax Analysis) และส่วนสร้างรหัส (Code Generater) โดยส่วนนี้จะทำหน้าที่แปลงจากรหัสชั่วคราวที่ได้มาจากขั้นตอนที่ 1 ไปเป็นรหัสภาษาเครื่องที่มีการทำงานสอดคล้องกัน จากนั้นก็จะทำการเติมค่าแอดเดรสลงไปในโปรแกรมในตำแหน่งที่มีการอ้างแอดเดรสโดยใช้ลาเบลโดยเราจะนำค่าแอดเดรสที่ตรงกับลาเบลนั้นมาใส่แทนที่ ดังนั้นจะเห็นได้ว่าเมื่อถึงจุดนี้เราก็จะได้ออบเจ็คโปรแกรมที่มีการทำงานที่ตรงกับโปรแกรมที่เขียนขึ้นทุกประการ

4.6.2 การพัฒนาโปรแกรมดีบัก (Debug)

ในการพัฒนาโปรแกรมนี้ ขั้นแรกก็ได้ศึกษาต้นแบบของโปรแกรมดีบักของซีพียูเบอร์อื่นๆ ซึ่งที่พบเห็นกันมากที่สุดก็คือโปรแกรม SYMDEB.EXE ของบริษัทไมโครซอฟท์ ซึ่งเป็นดีบักเกอร์ที่ใช้งานง่ายอีกทั้งมีขนาดที่ไม่ใหญ่จนเกินไป จากนั้นก็ได้นำโปรแกรมดีบักดังกล่าวมาเป็นต้นแบบในการพัฒนาโดยตัดส่วนที่ไม่จำเป็นออกไปบ้าง เพื่อประหยัดเวลาที่จะใช้ในการพัฒนาสำหรับดีบักที่ได้พัฒนาขึ้นนี้ก็มีฟังก์ชั่นการทำงานดังต่อไปนี้

A ใช้ในการทำแอสเซมเบลอร์

B ใช้ในการทำเบรคพอยท์ (Break Point) เพื่อให้โปรแกรมหยุดการทำงานตรงตำแหน่งที่เราต้องการ

D ใช้ในการดัมพ์ (Dump) เพื่อดูข้อมูลที่อยู่ในหน่วยความจำ

E ใช้ในการใส่ข้อมูลลงในหน่วยความจำ

F ใช้ในการใส่ข้อมูลลงในหน่วยความจำเช่นกัน แต่จะต่างที่ว่าในฟังก์ชันนี้จะใช้ในกรณีที่ต้องการใส่ข้อมูลที่มีค่าซ้ำๆ กันลงในหน่วยความจำเท่านั้น

G ใช้ในการสั่งให้โปรแกรมทำงาน

L ใช้ในการโหลดไฟล์ข้อมูลลงมาในหน่วยความจำ

M ใช้ในการเคลื่อนย้ายข้อมูลที่อยู่ในหน่วยความจำ

N ใช้ในการกำหนดชื่อไฟล์ที่จะอ่านหรือเขียน

Q ใช้ในการออกจากโปรแกรมนี้

R ใช้ในการดูข้อมูลที่เก็บไว้ในรีจิสเตอร์

S ใช้ในการค้นหาข้อมูลที่อยู่ในหน่วยความจำที่ต้องการ

T ใช้ทำการทำงานในโหมดซิงเกิ้ลสเต็ป (Single Step) ซึ่งจะเป็นโหมดที่มีการทำงานทีละคำสั่งเพื่อให้สามารถติดตามการทำงานได้อย่างใกล้ชิด

U ใช้ในการทำ อันแอสเซมเบลอร์ (Unasssemblar)

W ใช้ในการเขียนข้อมูลลงในดิสก์

! ใช้ในการทำเชลล์เพื่อที่จะสามารถเรียกโปรแกรมอื่นมาทำงานหรือสามารถเรียกการทำงานของดอสได้

? ใช้ในการดูข้อความแนะนำ จะเป็นประโยชน์มากเพราะสามารถแนะนำการใช้คำสั่งอย่างคร่าวๆ ได้

จากคำสั่งที่ตัวดีบักเกอร์นี้ทำได้ ก็จะเห็นได้ว่าโปรแกรมนี้จะเป็นประโยชน์อย่างมากต่อการพัฒนาโปรแกรม เพราะสามารถใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นได้

4.6.3 การพัฒนาโปรแกรมโหลดเดอร์

โปรแกรมนี้ทำหน้าที่เป็นเพียงโปรแกรมที่จะใช้ในการโหลดโปรแกรมที่ผ่านการแปลมาจากแอสเซมเบลอร์แล้ว ลงไปทำงานบนการ์ดที่สร้างขึ้น ดังนั้นจะเห็นได้ว่าเราจะใช้โปรแกรมนี้ก็ต่อเมื่อเราต้องการใช้โปรแกรมแอสเซมบลี้ที่เขียนขึ้นอย่างจริงจัง และนั่นก็หมายความว่าโปรแกรมนั้นควรจะเป็นโปรแกรมที่ได้ผ่านการทดสอบแล้วว่ามีการทำงานที่ถูกต้อง

4.7 <u>การทดสอบการทำงาน</u>

จากการทดสอบการทำงานของโปรแกรมทั้งสามแล้ว ก็มีผลการทำงานที่น่าพอใจ โปรแกรมสามารถทำงานได้ถูกต้อง คือโปรแกรมแอสเซมเบลอร์ก็สามารถทำงานได้และได้ผลลัพธ์จากการแปลถูกต้อง และโปรแกรมดีบักก็สามารถทำงานได้ดี อย่างไรก็ตามเนื่องจาก มีเวลาในการทดสอบไม่มากนัก จึงยังไม่พบข้อผิดพลาดมากนัก แต่ก็ได้ทำการแก้ไขเท่าที่พบ

บทที่ 5

บทวิจารณ์และสรุป

เนื่องจากไมโครโปรเซสเซอร์ 68000 นั้นเป็นไมโครโปรเซสเซอร์ ที่มีความสามารถสูงถ้าสามารถนำไปใช้ในการทำงานประมวลผลที่ต้องการความรวดเร็ว แต่เนื่องจากยังไม่มีเครื่องมือที่ช่วยเหลือ (tool) ในการพัฒนาระบบต่าง ๆ ที่จะนำเอาไมโครโปรเซสเซอร์ตัวนี้มาใช้ ดังนั้นโครงงานนี้ จึงเป็นการจัดสร้างระบบที่ช่วยเหลือในการพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์ 68000 เพื่ออำนวยความสะดวกในการที่จะนำไปใช้พัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์ 68000 เป็นตัวประมวลผลต่อไป

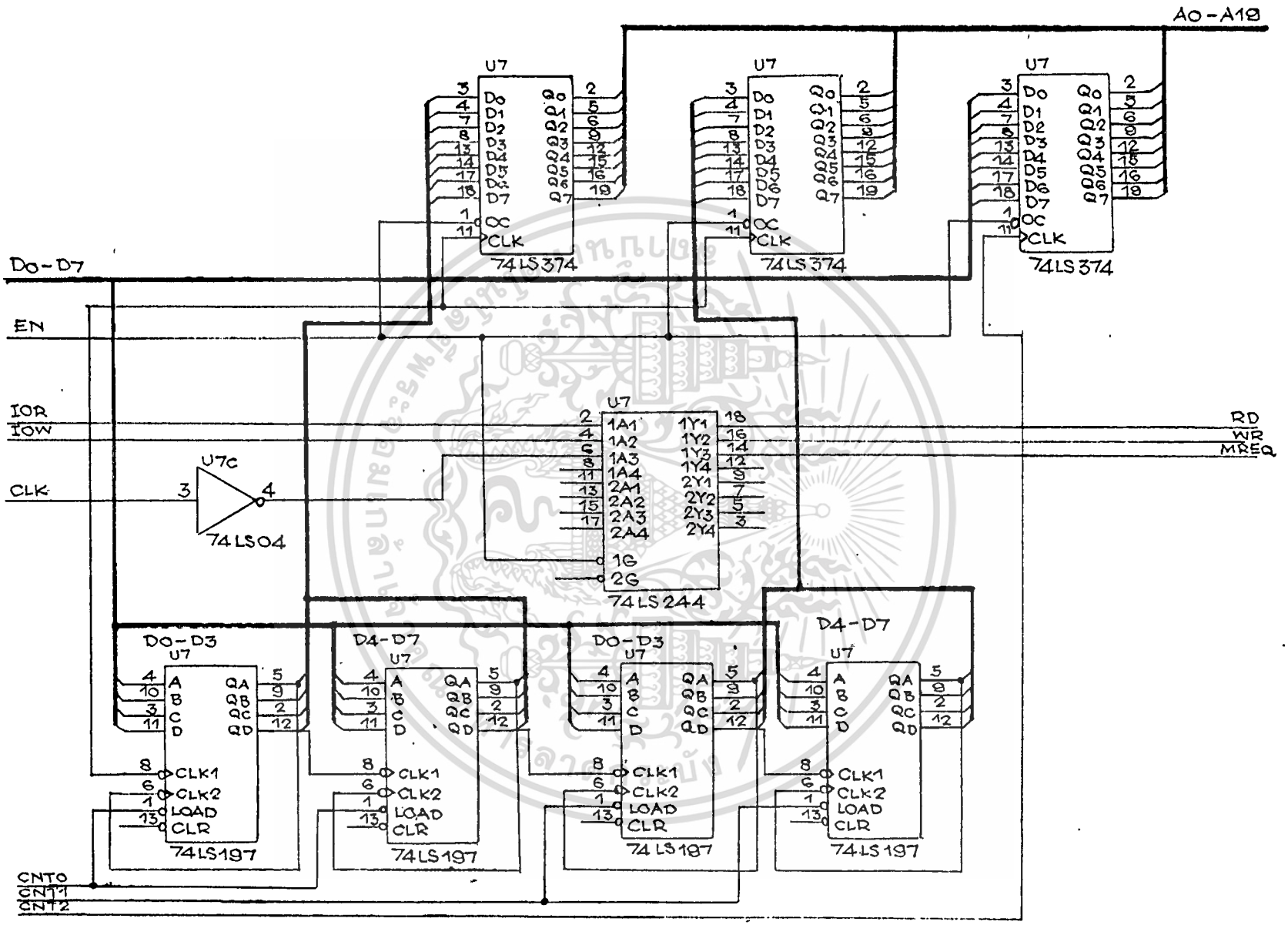
ในโครงงานนี้เราได้สร้างการ์ดขึ้นมาโดยใช้ MC 68000 เป็นซีพียู และมีวงจรอื่นๆ ประกอบด้วย เช่น หน่วยความจำ วงจรอินเตอร์เฟส วงจรควบคุมการทำดีเอ็มเอ (DMA Controller) วงจรมัลติเพล็กซ์ วงจรควบคุมบัสและบัฟเฟอร์ ซึ่งจะสามารถใช้การได้จริงโดยควบคุมการทำงานจาก ไอบีเอ็ม พีซี ดังนั้น ในโครงงานนี้จึงต้องเขียนส่วนซอฟท์แวร์บนเครื่องไอบีเอ็ม พีซี เพื่อทำงานร่วมกับวงจรบนการ์ดด้วย ซึ่งจะประกอบด้วยโปรแกรมแอสเซมเบลอร์ ดีบักเกอร์ และโหลดเดอร์

เนื่องจากตัวไมโครโปรเซสเซอร์ 68000 เป็นตัวที่ค่อนข้างใหม่และสลับซับซ้อนมาก ดังนั้นจึงต้องใช้เวลาในช่วงเทอมแรกทำการศึกษาการทำงานโดยละเอียด ก่อนที่จะทำการออกแบบวงจร ทำให้เสียเวลาในการทดลองและวินิจฉัยข้อผิดพลาด การสร้างการ์ดจะต้องสร้างวงจรย่อยต่างๆขึ้นมาประกอบกัน แต่ละวงจรย่อยจะต้องถูกออกแบบและสร้างขึ้นมาให้สัมพันธ์กับวงจรย่อยอื่นๆเพื่อให้การ์ดสามารถทำงานได้ ซึ่งจากการทดสอบนั้น ปรากฏว่าในส่วนวงจรติดต่อกับไอบีเอ็ม พีซีนั้นสามารถทำงานได้ โดยการทดสอบส่งข้อมูลออกไปทางพอร์ต และอ่านข้อมูลกลับเข้ามาตรวจสอบ และอีกวงจรอีกส่วนหนึ่งที่ทำการทดสอบ คือวงจรส่วนควบคุมการแลกเปลี่ยนข้อมูล ซึ่งการทดสอบวงจรส่วนนี้จะต้องเขียนโปรแกรมโหลดเดอร์ขึ้นมาก่อน และทดสอบโดยการลองส่งข้อมูลไปไว้ในหน่วยความจำบนการ์ด แล้วทดลองให้ส่งข้อมูลกลับมาตรวจสอบว่าถูกต้องหรือไม่ ซึ่งในครั้งแรกนั้นปรากฏว่าข้อมูลที่ทำการส่งไปแล้วอ่านกลับมานั้นมีค่าไม่ตรงจึงทำการตรวจแก้ไขในส่วนของหน่วยความจำ วงจรมัลติเพลกซ์ และส่วนควบคุมบัส จนสามารถแลกเปลี่ยนข้อมูลได้ถูกต้องเป็นส่วนมาก แต่ก็มีบางครั้งที่ยังมีปัญหาข้อมูลบางไบท์มีค่าไม่ตรง และเมื่อทดลองโหลดโปรแกรมที่เป็นออบเจคโค้ดของ 68000 ไปรัน ปรากฏว่ายังไม่สามารถทำงานให้ถูกต้องได้ ซึ่งจะต้องทำการตรวจสอบและแก้ไขต่อไป แต่เนื่องด้วยเวลาที่ทำโครงงานนี้

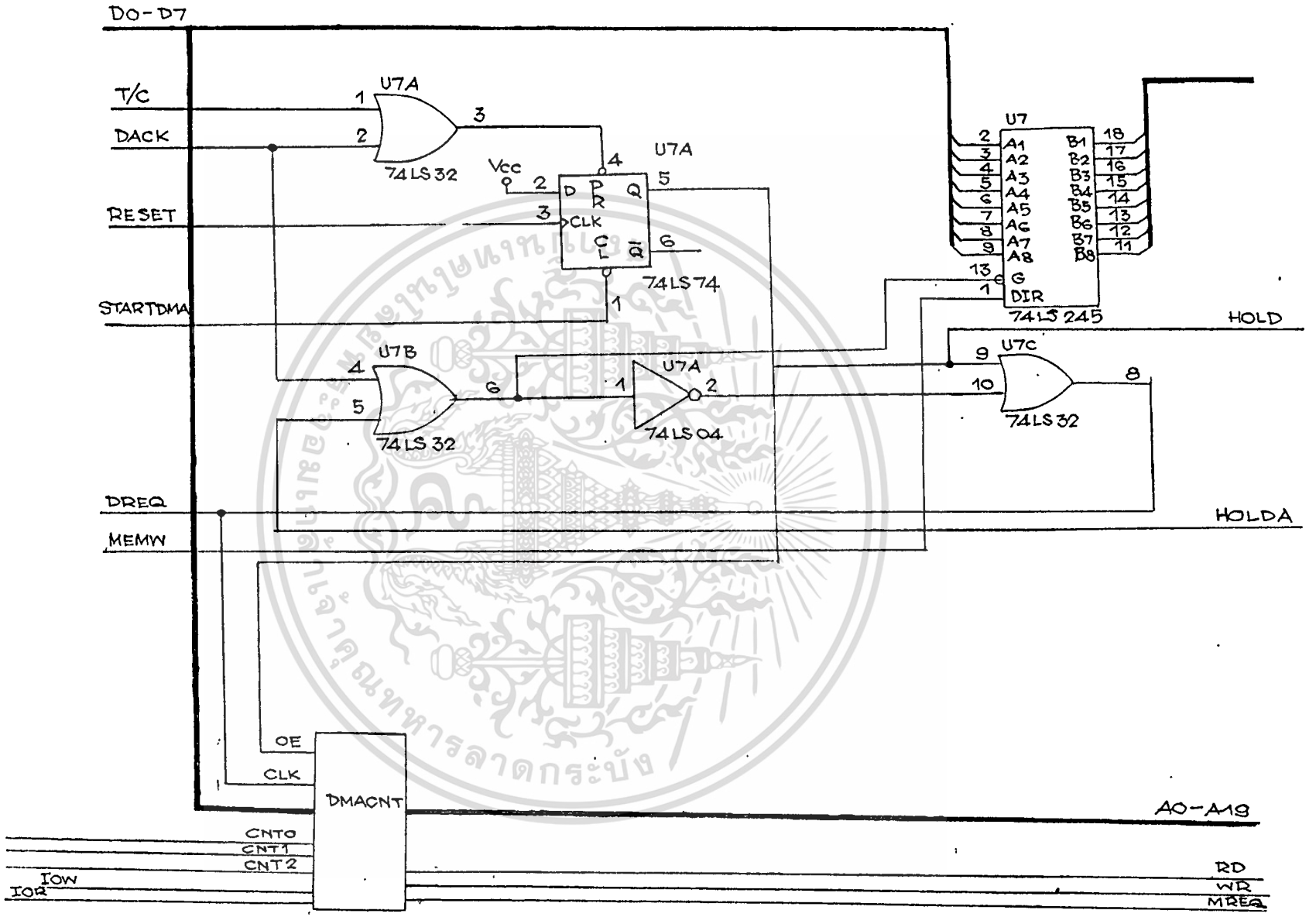
มีจำกัด และเสียเวลาในการศึกษาและตรวจแก้ไขในส่วนของฮาร์ดแวร์ จึงทำให้ผลงานในโครงงานนี้เสร็จไม่สมบูรณ์ตามเป้าหมายดีนัก

อย่างไรก็ตาม โครงงานนี้เป็นเพียงส่วนหนึ่ง ซึ่งสามารถใช้เป็นแนวทางในการพัฒนาระบบ ให้สมบูรณ์ยิ่งขึ้น เพราะจะช่วยแก้ไขปัญหาต่างๆข้างต้นได้ แต่การใช้งานจริงอาจไม่สะดวกนัก ดังนั้นโครงงานนี้ จึงต้องการ การพัฒนาตามรูปแบบ และจุดประสงค์ของผู้ต้องการใช้งาน

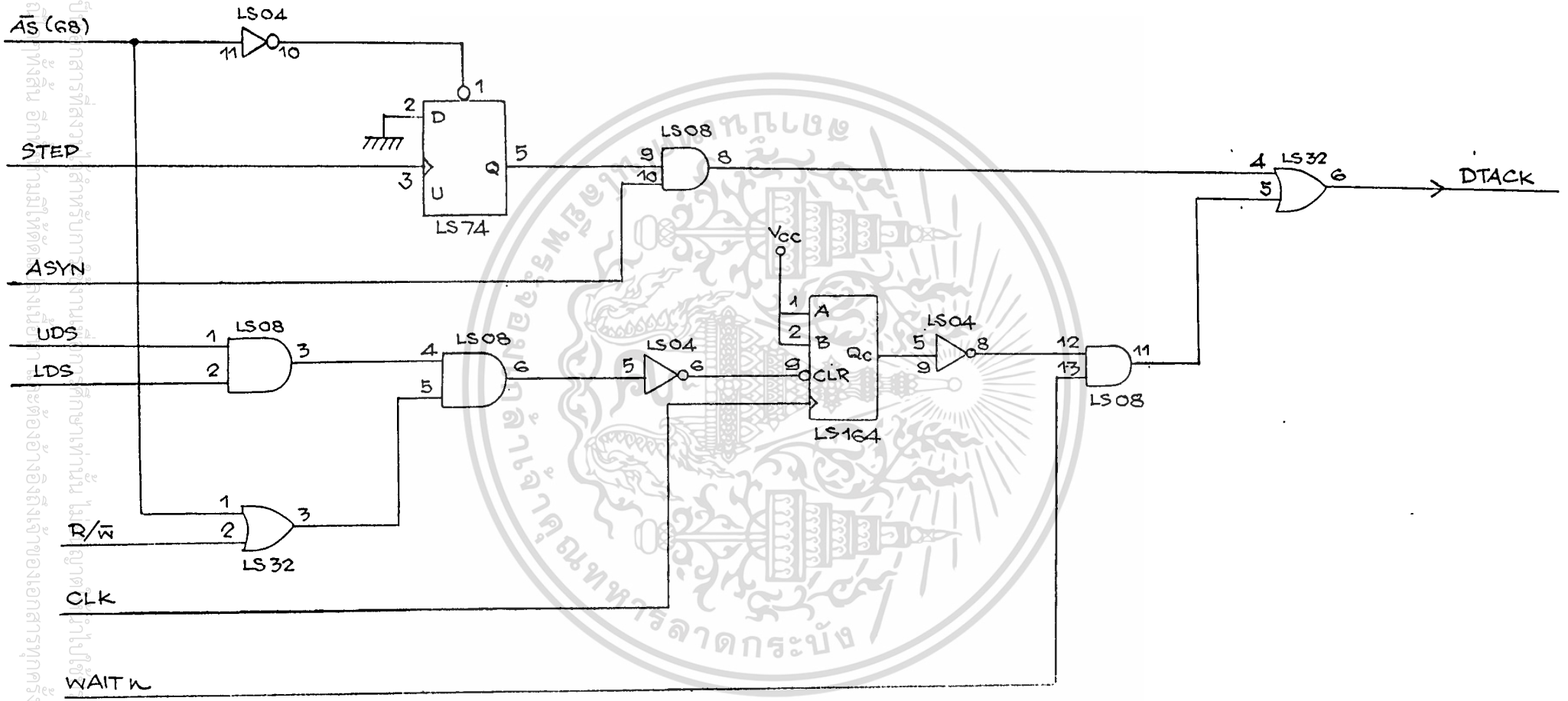
ภาคผนวก



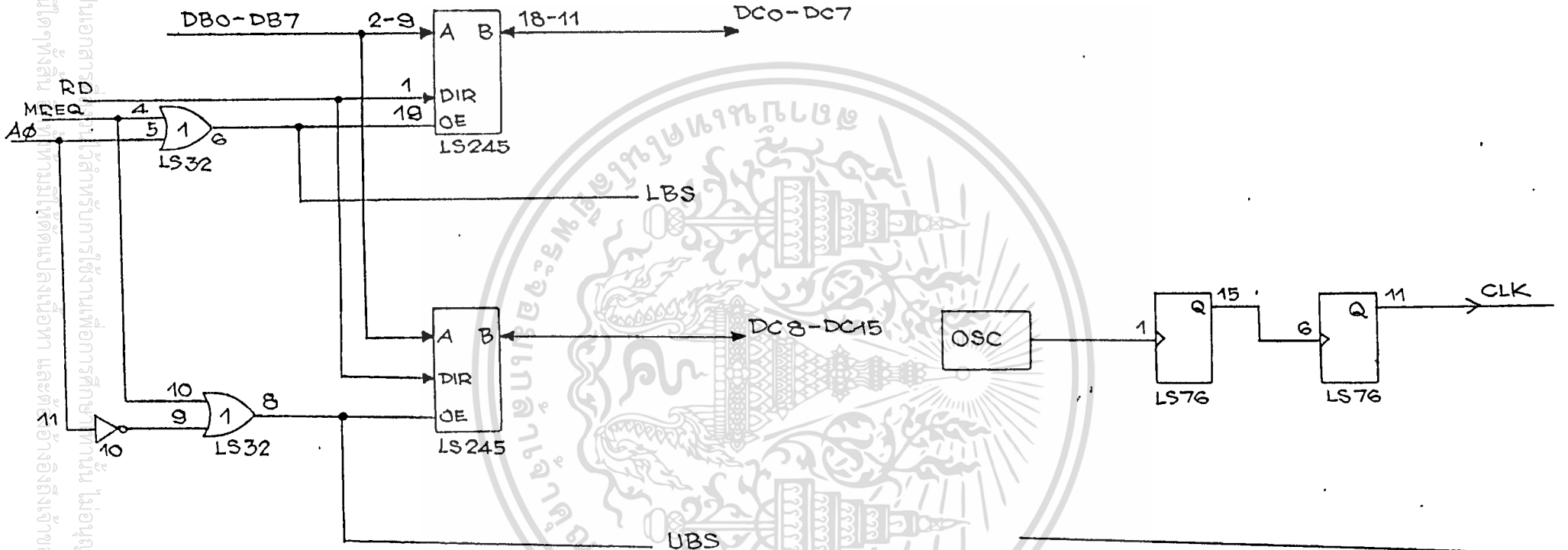
แสดงวงจร DMAC



DMAC INTERFACE TO IBM PC

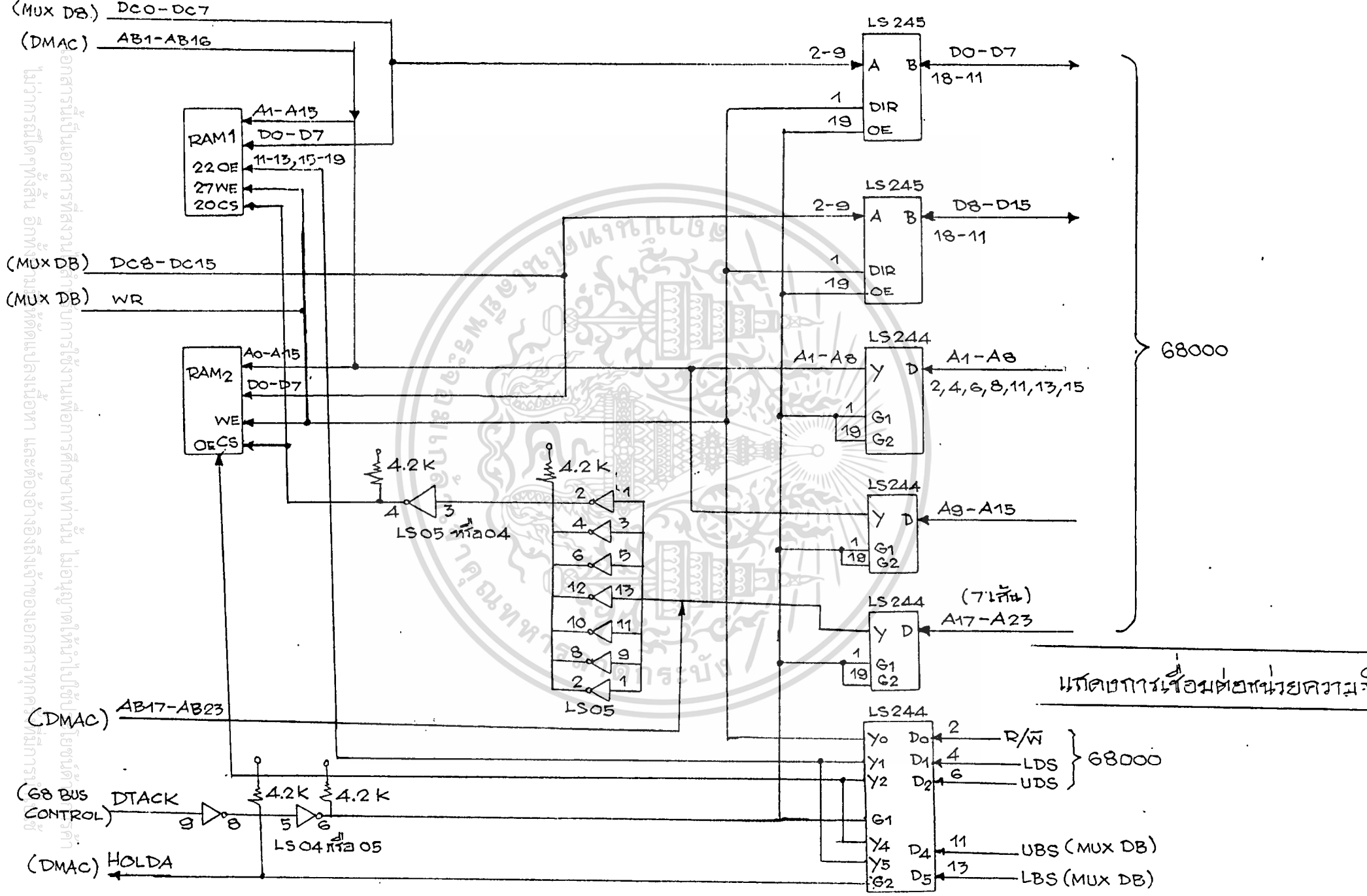


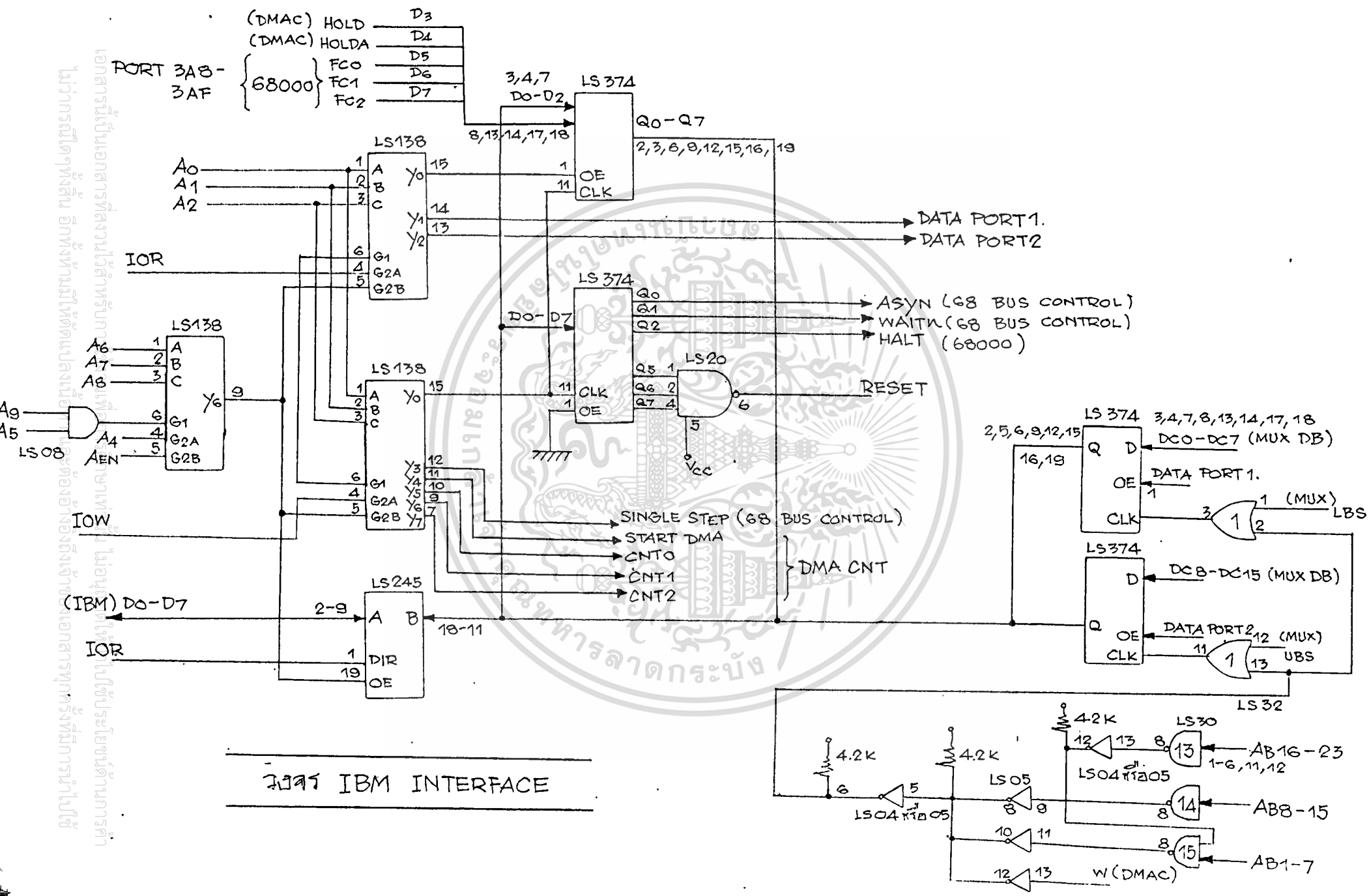
แสดงวงจร BUS CONTROL



แสดงวงจร MULTIPLEX

แสดงวงจรสร้างสัญญาณ CLOCK


(MUX DB) DC0-DC7
(DMAC) AB1-AB16
RAM1
A1-A15
D0-D7
11-13,15-19
22 OE
27 WE
20 CS
(MUX DB) DC8-DC15
(MUX DB) WR
RAM2
A0-A15
D0-D7
WE
OE CS
4.2 K
4.2 K
LS05 หรือ 04
LS05
LS245
2-9
A B
DIR
OE
1
19
D0-D7
18-11
LS245
D8-D15
LS244
A1-A8
Y D
G1
G2
2,4,6,8,11,13,15
LS244
A9-A15
LS244
(7 เส้น)
A17-A23
68000
(CDMAC) AB17-AB23
LS244
Y0 D0
Y1 D1
Y2 D2
G1
Y4 D4
Y5 D5
G2
R/W
LDS
UDS
UBS (MUX DB)
LBS (MUX DB)
(68 BUS CONTROL) DTACK
LS04 หรือ 05
(DMAC) HOLDA



วงจร IBM INTERFACE

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี ด้วยความร่วมมือจากหลายๆฝ่ายเป็นอย่างดี ผู้จัดทำขอขอบคุณ ภาควิชาคอมพิวเตอร์ ที่ได้เอื้อเฟื้อสถานที่ อุปกรณ์ และเครื่องไมโครคอมพิวเตอร์ พีซี ในการทำโครงงานและปริญญานิพนธ์

นอกจากนี้ ผู้จัดทำขอขอบคุณอย่างยิ่งสำหรับรุ่นพี่ เพื่อนๆ และรุ่นน้อง ภาควิชาคอมพิวเตอร์ ที่ให้คำแนะนำ ให้ข้อคิดเห็นอันเป็นประโยชน์ ช่วยเหลือชี้แนะแนวทางในการทำโครงงาน เอื้อเฟื้อหนังสือ พร้อมทั้งสถานที่ในการทำโครงงานนี้

จึงขอขอบคุณไว้ ณ โอกาสนี้

ผู้จัดทำ

หนังสืออ้างอิง

1. ประพันธ์ ธีระวรรณวิไล, "MC 68000 ไมโครโปรเซสเซอร์ 16/32 บิท", วารสารไมโครอิเล็คทรอนิคส์, ปีที่ 2, ฉบับที่ 9, 2527, หน้า 58-69

2. Alan D. Wilcox, Ph.D., P.E., "68000 Microcomputer System Designing and Troubleshooting", Prentice-Hall International, Inc., 579 P., 1987.

3. Alan R. Miller, "Assembly Language Techniques for IBM PC", Socorro, 358 P., 1986.

4. Brian W. Kernighan, Dennis M. Ritchie, "The C Programing Language", Prentice-Hall, INC., 228 P.

5. Lance A. Leventhal, Dong Hawkins, Gerry Kane, and William D. Cramer, "68000 Assembly Language Programming", Mc Graw_Hill, 484 P., 1986.

6. Microsoft Corporation, "Microsoft C Compiler Version 4.0", 444 P.

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

8. Motorola Inc., "MC 68000 16 BIT MICROPROCESSOR", APRIL, 1983.

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

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

11. Steve Schustack, "Variations in C", Microsoft Press, 344 P., 1985.