บทที่ 3

การออกแบบโปรแกรมควบคุมระบบ

3.1 หน้าที่ของโปรแกรมควบคุมระบบ สำหรับเครื่องไมโครคอมพิวเตอร์ของสถาบันบริการคอมพิวเตอร์ จุฬาฯ

3.1.1 สามารถรับการเปลี่ยนแปลงคำสั่ง (COMMAND) ที่ใช้ในการติดต่อกับโปรแกรมควบคุมระบบได้

3.1.2 สามารถรับคำสั่งจากผู้ใช้ได้

3.1.3 สามารถทำการติดต่อกับจานแม่เหล็ก, เครื่องพิมพ์, แป้นพิมพ์ และจอภาพได้

3.1.4 เฟิร์มแวร์ของระบบสามารถอ่านโปรแกรมควบคุมระบบ ไปไว้ในหน่วยความจำได้

3.2 การออกแบบวิธีการติดต่อระหว่างผู้ใช้กับโปรแกรมควบคุมระบบ

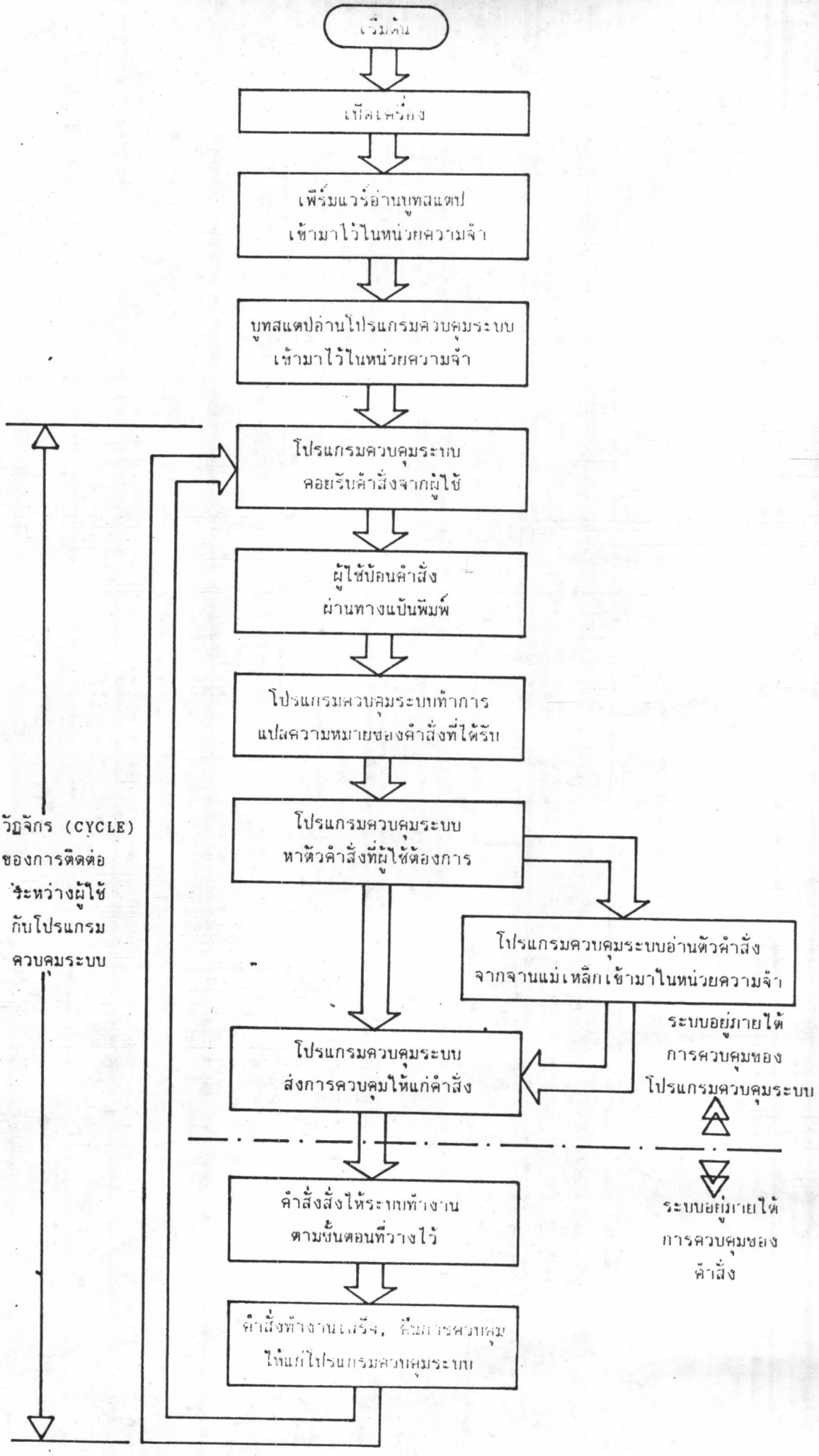
ผู้ใช้ต้องสามารถติดต่อกับโปรแกรมควบคุมระบบได้ เพื่อสั่งให้ระบบทำงานตามที่ต้องการ การออกแบบวิธีที่ผู้ใช้ใช้ติดต่อกับโปรแกรมควบคุมระบบ จะอยู่ในลักษณะที่ ผู้ใช้ป้อนคำสั่งผ่านทางแป้นพิมพ์ ซึ่งโปรแกรมควบคุมระบบจะรอรับคำสั่ง เพื่อนำไปแปลความหมาย (INTERPRET) แล้วทำงานตามคำสั่งนั้น และทุกครั้งที่ทำคำสั่งหนึ่งเสร็จ โปรแกรมควบคุมระบบจะต้องมารอรับคำสั่งต่อไปจากผู้ใช้เสมอ ซึ่งขั้นตอนจะวนเป็นวัฏจักร (CYCLE) ไปเรื่อยๆ ดังนั้นการทำงานของระบบจะสิ้นสุด ก็ต่อเมื่อผู้ใช้ทำการปิดเครื่อง

ในรูปที่ 3.1 แสดงขั้นตอนในการติดต่อ ซึ่งมีรายละเอียดดังต่อไปนี้

ขั้นที่ 1 ผู้ใช้ทำการเปิดเครื่อง

ขั้นที่ 2 เฟิร์มแวร์ซึ่งอยู่ในรอมของระบบ ทำการอ่านบูทสแต็ปจากจานแม่เหล็กเข้ามาในหน่วยความจำ

ขั้นที่ 3 บูทสแต็ปทำการอ่านโปรแกรมควบคุมระบบ จากจานแม่เหล็ก



รูปที่ 3.1 ขั้นตอนการติดต่อระหว่างผู้ใช้กับโปรแกรมควบคุมระบบ

เข้ามาไว้ในหน่วยความจำ

ขั้นที่ 4 โปรแกรมควบคุมระบบคอยรับคำสั่งจากผู้ใช้

ขั้นที่ 5 ผู้ใช้ป้อนคำสั่งผ่านแป้นพิมพ์

ขั้นที่ 6 โปรแกรมควบคุมระบบทำการแปลความหมายของคำสั่ง

ขั้นที่ 7 โปรแกรมควบคุมระบบหาตัวคำสั่งที่ผู้ใช้ต้องการ

ขั้นที่ 8 ถ้าคำสั่งนั้น เป็นคำสั่งประจำซึ่งมีอยู่แล้วในตัวโปรแกรมควบคุมระบบ โปรแกรมควบคุมระบบจะส่งการควบคุมให้แก่คำสั่งนั้นทันที

ขั้นที่ 9 ถ้าคำสั่งนั้น เป็นคำสั่งที่เก็บอยู่บนจานแม่เหล็ก โปรแกรมควบคุมระบบจะทำการอ่านตัวคำสั่งนั้น เข้ามาไว้ในหน่วยความจำก่อน เสร็จแล้วจึงส่งการควบคุมให้แก่คำสั่งนั้น

ขั้นที่ 10 คำสั่งสั่งให้ระบบทำงาน ตามขั้นตอนที่วางไว้

ขั้นที่ 11 เมื่อคำสั่งทำงานเสร็จ จะสั่งการควบคุมคืนให้แก่โปรแกรมควบคุมระบบ ซึ่งโปรแกรมควบคุมระบบจะคอยรอรับคำสั่งต่อไปจากผู้ใช้ ตามขั้นที่ 4

3.3 การทำงานโดยโปรแกรมควบคุมระบบ

การที่โปรแกรมควบคุมระบบจะทำงานตามคำสั่งของผู้ใช้ได้ โปรแกรมควบคุมระบบจะต้องถูกออกแบบ ให้มีความสามารถในการทำงานต่างๆ ดังต่อไปนี้

3.3.1 การทำงานเกี่ยวกับการควบคุมระบบ

3.3.1.1 ทำการรับคำสั่งจากผู้ใช้

3.3.1.2 ทำการแปลความหมายของคำสั่งที่ได้รับ

3.3.1.3 ทำการส่งการควบคุมระบบให้แก่คำสั่ง

3.3.1.4 ทำการรับการควบคุมระบบคืนจากคำสั่ง

3.3.2 การทำงานเกี่ยวกับหน่วยความจำ

3.3.2.1 ทำการจัดพื้นที่หน่วยความจำ สำหรับคำสั่งที่ผู้ใช้ต้องการ

3.3.2.2 ทำหน้าที่เป็นโหลดเดอร์ (LOADER) อ่านคำสั่งจากจานแม่เหล็กลงไว้ในหน่วยความจำ

3.3.3 การทำงานเกี่ยวกับการติดต่อกับอุปกรณ์

3.3.3.1 ทำการรับข้อมูลจากแป้นพิมพ์

3.3.3.2 ทำการส่งข้อมูลออกจอภาพ

3.3.3.3 ทำการส่งข้อมูลออกเครื่องพิมพ์

3.3.3.4 ทำการอ่าน/บันทึกข้อมูลกับจานแม่เหล็ก

3.3.4 การจัดการเกี่ยวกับแฟ้มข้อมูล

3.3.4.1 ทำการจัดระบบการเก็บแฟ้มข้อมูล

3.3.4.2 ทำการสร้างและลบแฟ้มข้อมูล

3.3.4.3 ทำการเปิดและปิดแฟ้มข้อมูล

3.3.4.4 ทำการอ่าน/บันทึกข้อมูลของแฟ้มทั้งแบบเรียงลำดับและแบบสุ่ม

3.4 การออกแบบฐานข้อมูล

ในการออกแบบโปรแกรมควบคุมระบบ จะต้องทำการกำหนดพื้นที่ส่วนหนึ่งสำหรับใช้เป็นฐานข้อมูล เพื่อใช้สำหรับเก็บ, รับ และส่งข้อมูลที่เกิดขึ้นในระหว่างการทำงานของระบบ ในการออกแบบนี้จะกำหนดพื้นที่ 2 ชนิด เพื่อใช้เป็นฐานข้อมูล คือ

3.4.1 กำหนดหน่วยความจำส่วนหนึ่งเป็นบัฟเฟอร์ สำหรับใช้เก็บข้อมูล

3.4.2 ใช้รีจิสเตอร์ของซีพียู เป็นที่เก็บข้อมูล

การกำหนดพื้นที่ชนิดใดเป็นฐานข้อมูลของแต่ละโปรแกรม พิจารณาจากความเหมาะสม, ความเป็นไปได้ และข้อดี/ข้อเสีย ดังต่อไปนี้

การใช้บัฟเฟอร์เป็นฐานข้อมูล

ข้อดี 1. มีขนาดได้ไม่จำกัด

2. ข้อมูลที่เก็บไม่สูญหาย สามารถใช้ร่วมได้หลายๆโปรแกรม

ข้อเสีย 1. ใช้เนื้อที่หน่วยความจำ ทำให้โปรแกรมมีขนาดใหญ่ขึ้น

2. การกำหนดมาตรฐาน ในการติดต่อระหว่างโปรแกรมทำได้ยาก ตำแหน่งของบัฟเฟอร์ต้องมีแอดเดรสที่แน่นอน และทุกโปรแกรมที่ใช้ข้อมูลในบัฟเฟอร์นั้น ต้องรับรู้แอดเดรสนี้

การใช้รีจิสเตอร์เป็นฐานข้อมูล

ข้อดี 1. ไม่ใช้เนื้อที่หน่วยความจำ จึงไม่ทำให้ขนาดของโปรแกรมเพิ่ม

2. การกำหนดมาตรฐานในการติดต่อระหว่างโปรแกรมทำได้สะดวก

3. สามารถจัดโปรแกรมให้เป็นโมดุลได้สะดวก

ข้อเสีย 1. มีขนาดจำกัด ข้อมูลมีขนาดเพียง 1 ไบต์ สำหรับการใช้รีจิสเตอร์เดี่ยว และมีขนาดสูงสุด 2 ไบต์ สำหรับการใช้รีจิสเตอร์คู่ (รายละเอียดของรีจิสเตอร์ ดูภาคผนวก ก)

2. การใช้ข้อมูลร่วมกันหลายๆ โปรแกรมทำได้ยาก ข้อมูลมักสูญหายในระหว่างการทำงาน

3.5 การออกแบบโปรแกรมควบคุมระบบ

จากวิธีการติดต่อกับผู้ใช้ในหัวข้อ 3.2 จากการทำงานต่างๆที่ทำโดยโปรแกรมควบคุมระบบในหัวข้อ 3.3 และจากลักษณะของฐานข้อมูลในหัวข้อ 3.4 ทำให้ทราบถึงลักษณะของข้อมูลเข้า (INPUT), การทำงาน (FUNCTION) และฐานข้อมูลของโปรแกรมควบคุมระบบที่จะสร้างขึ้น ซึ่งสิ่งเหล่านี้เป็นข้อมูลที่ใช้ในการออกแบบโปรแกรมควบคุมระบบ

การออกแบบโปรแกรมควบคุมระบบ จะออกแบบให้มีลักษณะเป็นหลายๆ โมดุลประกอบเข้าด้วยกันเป็นระบบ โดยแต่ละโมดุลเกิดจากการรวมเอาโปรแกรมที่มีการทำงานคล้ายๆ กัน และมีการใช้ฐานข้อมูลร่วมกันเข้าไว้ด้วยกัน เหตุผลที่ออกแบบให้เป็นโมดุล ก็เพื่อความสะดวกในการเขียน (CODING), การทดสอบ (TESTING) และการหาความผิดพลาด (DEBUGING) เพราะสามารถทำทีละโมดุลแยกจากกัน ไม่จำเป็นต้องทำพร้อมกันทั้งระบบ อีกทั้งการปรับปรุงโปรแกรมในภายหลังทำได้ง่ายเพราะทำเพียงเฉพาะโมดุล โดยไม่มีผลกระทบไปยังโมดุลอื่น

โปรแกรมควบคุมระบบนี้ ถูกออกแบบให้ประกอบด้วยโมดุล ดังต่อไปนี้คือ

3.5.1 โมดุล I ทำหน้าที่เกี่ยวกับการควบคุมระบบ (หัวข้อ 3.3.1) และการจัดการหน่วยความจำ (หัวข้อ 3.3.2)

3.5.2 โมดุล II ทำหน้าที่เกี่ยวกับแฟ้มข้อมูลทั้งหมด (หัวข้อ 3.3.4)

3.5.3 โมดุล III ทำหน้าที่เกี่ยวกับการติดต่อกับอุปกรณ์ (หัวข้อ 3.3.3)

3.3.4 โมดุล IV เป็นเฟิร์มแวร์ที่ใช้ติดต่อกับฮาร์ดแวร์ของอุปกรณ์ โมดุลนี้เกิดขึ้นจากการออกแบบทางด้านฮาร์ดแวร์ของระบบ

3.6 ความสัมพันธ์ระหว่างโมดุล

จากการออกแบบโปรแกรมควบคุมระบบ ให้มีลักษณะเป็นโมดุล ซึ่งแต่ละโมดุลต่างก็ทำหน้าที่เฉพาะอย่าง ดังนั้นเมื่อโมดุลหนึ่งถูกสั่งให้ทำงานซึ่งตนไม่สามารถทำได้ จะให้โมดุลนั้นส่งต่อข้อมูลพร้อมกับสั่งให้โมดุลอื่นทำงานแทน เพื่อให้ผลสุดท้ายได้ผลลัพธ์ หรือเกิดการทำงานตามที่ต้องการ นั่นคือในการทำงานของระบบ เกิดจากการทำงานประสานกันระหว่างโมดุลต่างๆ ซึ่งจะทำเช่นนั้นได้จะต้องมีการกำหนดขั้นตอนที่ใช้ในการสั่งงาน, การส่งข้อมูล และการรับผลลัพธ์ระหว่างโมดุลหนึ่งกับอีกโมดุลหนึ่ง

ในการออกแบบ จะกำหนดความสัมพันธ์ระหว่างโมดุลต่างๆ โดยแบ่งออกเป็นระดับ เริ่มตั้งแต่ระดับผู้ใช้ ดังแสดงในรูปที่ 3.2 ซึ่งรายละเอียดมีดังนี้

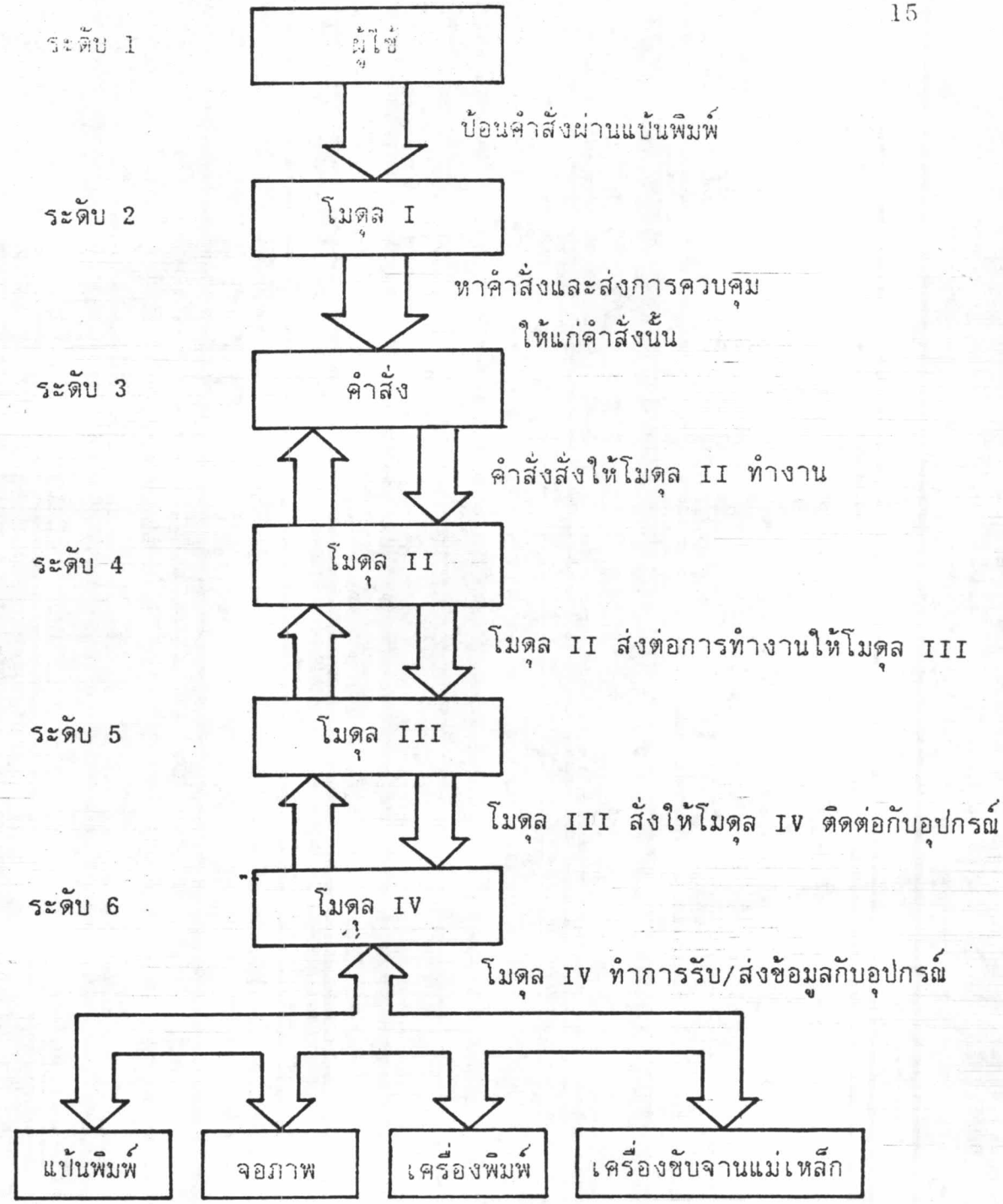
ระดับ 1 ผู้ใช้ ผู้ใช้ทำการป้อนคำสั่งผ่านแป้นพิมพ์ ซึ่งโมดุล I จะรอรับคำสั่งนี้

ระดับ 2 โมดุล I โมดุล I ทำการแปลความหมายของคำสั่งที่ได้รับเพื่อหาว่าผู้ใช้ต้องการใช้คำสั่งใด จากนั้นจะส่งการควบคุมระบบให้แก่คำสั่งนั้น

ระดับ 3 คำสั่ง คำสั่งสั่งให้ระบบทำงานตามขั้นตอนที่วางไว้ โดยสั่งผ่านโมดุล II พร้อมกับส่งข้อมูลไปให้ ซึ่งเมื่อโมดุล II ทำงานเสร็จก็จะส่งผลลัพธ์กลับคืนมา

ระดับ 4 โมดุล II โมดุล II ได้รับการสั่งงานและข้อมูลจากคำสั่งก็จะเอาข้อมูลนั้นไปใช้งาน เสร็จแล้วส่งผลลัพธ์ที่ได้กลับไปให้คำสั่ง โดยถ้าการทำงานนั้น เป็นการทำงานเกี่ยวกับแฟ้มข้อมูล การทำงานจะอยู่เฉพาะภายในโมดุล II แต่ถ้าเป็นการทำงานเกี่ยวกับการรับ/ส่งข้อมูลกับอุปกรณ์ โมดุล II จะส่งต่อข้อมูลให้โมดุล III ทำงานแทน และเมื่อโมดุล III ส่งผลลัพธ์กลับมาโมดุล II จะส่งต่อผลลัพธ์กลับไปให้คำสั่ง ในกรณีนี้โมดุล II ทำหน้าที่เป็นทางผ่านในการติดต่อระหว่างคำสั่งกับโมดุล III

ระดับ 5 โมดุล III เมื่อโมดุล III ได้รับข้อมูลที่ส่งต่อมาจากโมดุล II ก็จะทำการรับ/ส่งข้อมูลนั้นกับอุปกรณ์ที่ต่ออยู่กับระบบ แล้วส่งผลลัพธ์



รูปที่ 3.2 ความสัมพันธ์ระหว่างโมดุล

กลับไปให้แก่โมดุล II ในการทำงานของโมดุล III จะอาศัยการทำงานทางด้านฮาร์ดแวร์ของโมดุล IV อีกต่อหนึ่ง

ระดับ 6 โมดุล IV เป็นการทำงานขั้นสุดท้าย โดยโมดุล IV นี้เป็นเฟิร์มแวร์ของระบบ สำหรับทำหน้าที่ติดต่อทางฮาร์ดแวร์กับอุปกรณ์ โดยทำการส่งข้อมูลให้แก่อุปกรณ์(กรณีทำ OUTPUT) หรือรับข้อมูลจากอุปกรณ์(กรณีทำ INPUT) พร้อมกับส่งข้อมูลนั้นให้แก่โมดุล III

3.7 ความสัมพันธ์ระหว่างฐานข้อมูล

ในการทำงานของระบบ นอกจากจะต้องมีความสัมพันธ์ในการทำงานระหว่างโมดุลแล้ว ยังต้องมีความสัมพันธ์กันของฐานข้อมูลด้วย เพราะในการส่งค่าระหว่างโมดุล จะทำผ่านฐานข้อมูลเหล่านี้ โดยฐานข้อมูลที่ใช้ได้แก่บัฟเฟอร์และรีจิสเตอร์ต่างๆ ความสัมพันธ์ของฐานข้อมูลที่ใช้ในการติดต่อระหว่างโมดุลกำหนดไว้ดังแสดงในรูปที่ 3.3 รายละเอียดมีดังนี้

3.7.1 บัฟเฟอร์ของแป้นพิมพ์ ใช้เนื้อที่หน่วยความจำของโมดุล I ขนาด 128 ไบต์ สำหรับเก็บคำสั่งที่ผู้ใช้ป้อนเข้ามาทางแป้นพิมพ์

3.7.2 บัฟเฟอร์ของคำสั่ง ใช้สำหรับเก็บค่าพารามิเตอร์ที่คำสั่งต้องการใช้โดยอ่านมาจากบัฟเฟอร์ของแป้นพิมพ์ พื้นที่ส่วนนี้อยู่ในตัวโปรแกรมคำสั่งเอง

3.7.3 รีจิสเตอร์ติดต่อกับโมดุล II การติดต่อกับโมดุล II จะทำผ่านรีจิสเตอร์ต่างๆ ดังต่อไปนี้

3.7.3.1 การส่งข้อมูลให้โมดุล

รีจิสเตอร์ C = ชนิดของงานที่สั่งให้โมดุลทำ

รีจิสเตอร์ E = ข้อมูล (กรณีที่ข้อมูลที่มีขนาด 1 ไบต์)

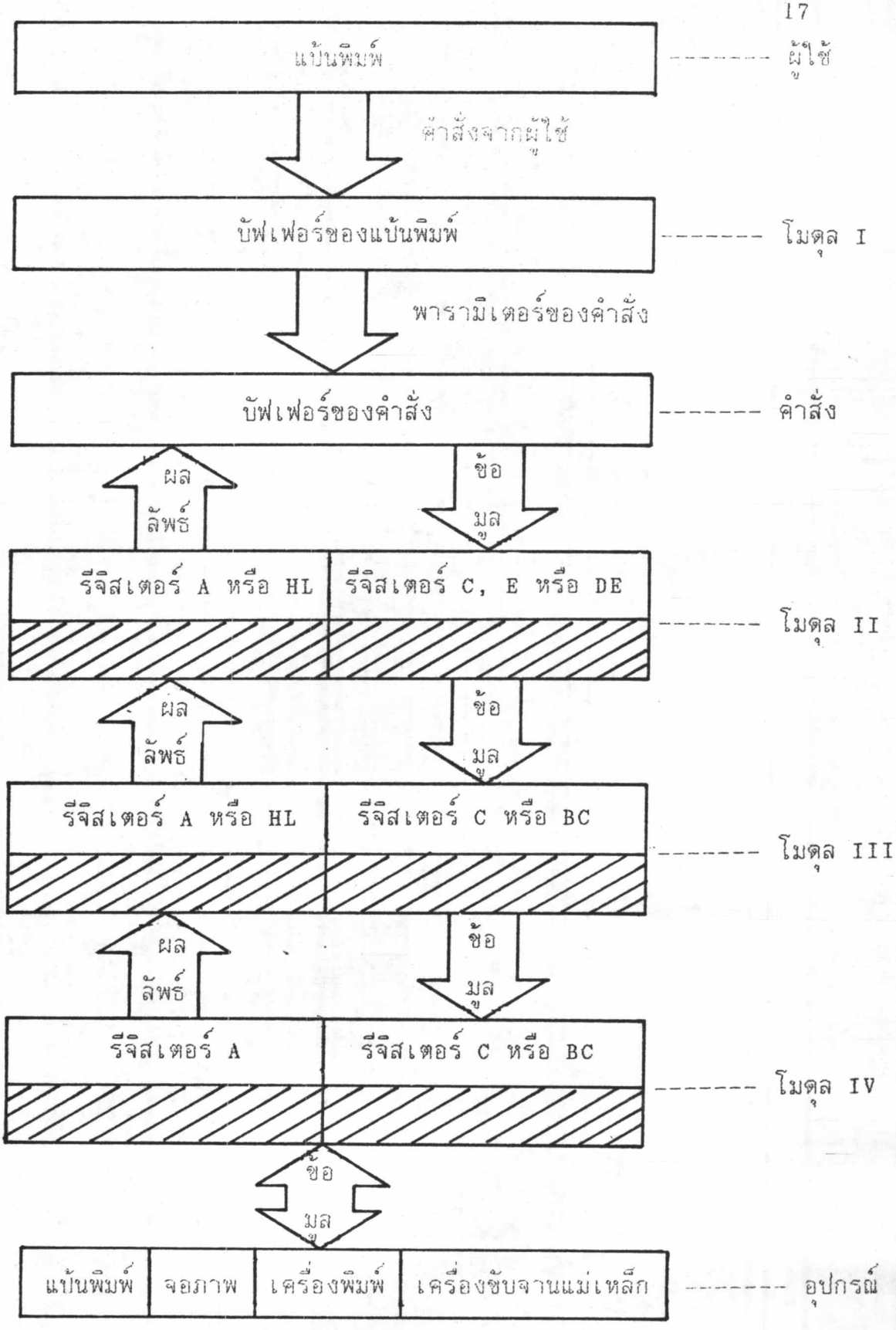
หรือ รีจิสเตอร์ DE = ข้อมูล (กรณีข้อมูลที่มีขนาด 2 ไบต์)

3.7.3.2 การรับผลลัพธ์จากโมดุล

รีจิสเตอร์ A = ผลลัพธ์ (กรณีที่ผลลัพธ์มีขนาด 1 ไบต์)

หรือ รีจิสเตอร์ HL = ผลลัพธ์ (กรณีที่ผลลัพธ์มีขนาด 2ไบต์)

3.7.4 รีจิสเตอร์ติดต่อกับโมดุล III การติดต่อกับโมดุล III จะทำผ่านรีจิสเตอร์ต่างๆ ดังต่อไปนี้



รูปที่ 3.3 ความสัมพันธ์ระหว่างฐานข้อมูล

3.7.4.1 การส่งข้อมูลให้โมดุล

รีจิสเตอร์ C = ข้อมูล (กรณีที่ข้อมูลมีขนาด 1 ไบต์)

หรือ รีจิสเตอร์ BC = ข้อมูล (กรณีที่ข้อมูลมีขนาด 2 ไบต์)

3.7.4.2 การรับผลลัพธ์จากโมดุล

รีจิสเตอร์ A = ผลลัพธ์ (กรณีที่ผลลัพธ์มีขนาด 1 ไบต์)

หรือ รีจิสเตอร์ HL = ผลลัพธ์ (กรณีที่ผลลัพธ์มีขนาด 2 ไบต์)

3.7.5 รีจิสเตอร์ติดต่อกับโมดุล IV การติดต่อกับโมดุล IV จะทำผ่านรีจิสเตอร์ต่างๆ ดังต่อไปนี้

3.7.5.1 การส่งข้อมูลให้โมดุล

รีจิสเตอร์ C = ข้อมูล (กรณีที่ข้อมูลมีขนาด 1 ไบต์)

หรือ รีจิสเตอร์ BC = ข้อมูล (กรณีที่ข้อมูลมีขนาด 2 ไบต์)

3.7.5.2 การรับข้อมูลจากโมดุล

รีจิสเตอร์ A = ผลลัพธ์

3.8 การออกแบบโมดุล

ในการออกแบบโมดุล จะจัดให้แต่ละโมดุลทำหน้าที่ ตามที่แบ่งไว้ในหัวข้อ 3.5.1-3.5.4 และในการติดต่อระหว่างโมดุล ใช้ฐานข้อมูลตามที่กำหนดไว้ในหัวข้อ 3.7

3.8.1 การออกแบบโมดุล I โมดุลนี้จะตั้งชื่อเรียกว่า โมดุลอาร์พีเอ (RPA หรือ RESIDENT PROGRAM AREA) ซึ่งออกแบบให้มีลักษณะและการทำงานดังต่อไปนี้

3.8.1.1 ทำการติดต่อกับผู้ใช้

- โมดุลอาร์พีเอ เป็นโมดุลที่ระบบใช้ติดต่อกับผู้ใช้
- ผู้ใช้ติดต่อกับโมดุลอาร์พีเอ โดยป้อนบันทัดคำสั่ง (COMMAND LINE) ผ่านทางแป้นพิมพ์
- โมดุลอาร์พีเอจะใช้เนื้อที่หน่วยความจำขนาด 128 ไบต์ เป็นบัฟเฟอร์ของแป้นพิมพ์
- โมดุลอาร์พีเอจะรับคำสั่งจากผู้ใช้ มาเก็บไว้ที่บัฟเฟอร์ของแป้นพิมพ์

3.8.1.2 ทำหน้าที่เป็น LINE INTERPRETER

- โมดุลอาร์พีเอ ทำหน้าที่แปลความหมายของบันทัดคำสั่ง ที่รับมาจากผู้ใช้

- การแปลบันทัดคำสั่ง ยึดรูปแบบดังนี้

วรรคแรกของบันทัดคำสั่ง เป็นชื่อของคำสั่งที่ผู้ใช้ต้องการใช้ และวรรคต่อๆ มาเป็นพารามิเตอร์ของคำสั่งนั้น ตัวอย่างเช่น

A>DEBUG TESTPROG.Z80 <RET>

หมายความว่า ผู้ใช้ต้องการใช้คำสั่งชื่อ "DEBUG" ซึ่งจะทำงานกับแฟ้มข้อมูลชื่อ "TESTPROG.Z80"

3.8.1.3 ทำการรับและส่งการควบคุมระบบ

- เมื่อหาคำสั่งที่ผู้ใช้ต้องการใช้พบ โมดุลอาร์พีเอจะส่งการควบคุมระบบให้แก่คำสั่งนั้น

- รับการควบคุมระบบกลับคืนจากคำสั่ง เมื่อคำสั่งนั้นทำงานเสร็จ

3.8.1.4 ทำหน้าที่เป็นโหลดเดอร์ (LOADER)

- กรณีคำสั่งที่ต้องการใช้ เป็นคำสั่งที่เก็บไว้บนจานแม่เหล็ก โมดุลอาร์พีเอจะทำหน้าที่เป็นโหลดเดอร์ อ่านเอาคำสั่งนั้นเข้ามาไว้ในหน่วยความจำ

- การอ่านคำสั่งเข้ามาไว้ในหน่วยความจำ โมดุลอาร์พีเอจะเป็นผู้กำหนดโหลดแอดเดรสให้

3.8.1.5 เป็นที่เก็บคำสั่งประจำของระบบ

- ส่วนหนึ่งของโมดุลอาร์พีเอ เป็นที่เก็บคำสั่ง 9 คำสั่ง
- คำสั่งเหล่านี้เก็บเอาไว้ประจำ(RESIDENT)ในระบบ
- แต่ละคำสั่งผู้ใช้สามารถเรียกใช้ได้เสมอ

จากการออกแบบโมดุลอาร์พีเอนี้ จะเห็นได้ว่าคำสั่งที่ผู้ใช้เรียกใช้แบ่งออกเป็น 2 พวก คือ คำสั่งประจำของระบบซึ่งเก็บอยู่ในโมดุลอาร์พีเอ และคำสั่งที่เก็บไว้บนจานแม่เหล็ก เหตุผลที่เลือกออกแบบให้มีลักษณะเช่นนี้ก็เพื่อ

- ความรวดเร็วในการทำงาน เพราะถ้าคำสั่งที่ผู้ใช้ต้องการเป็นคำสั่งประจำ การทำงานจะเกิดขึ้นทันที

- ความสามารถในการเพิ่มคำสั่งที่ใช้ในระบบ เพราะถ้าคำสั่งประ-

จำที่มีอยู่ไม่เพียงพอต่อการใช้งาน ผู้ใช้สามารถเขียนคำสั่งเพิ่มขึ้น แล้วเก็บไว้บนจานแม่เหล็ก ถึงเวลาต้องการใช้จึงเรียกมาใช้งาน ด้วยวิธีนี้ทำให้คำสั่งที่ใช้กับระบบสามารถเพิ่มเติมได้ไม่มีขีดจำกัด

ในการตั้งชื่อโมดุลนี้ ตั้งขึ้นตามลักษณะของโมดุลคือเป็นที่เก็บคำสั่งส่วนหนึ่งไว้ประจำ (RESIDENT)

3.8.2 การออกแบบโมดุล II โมดุลนี้จะตั้งชื่อเรียกว่า โมดุลดอส (DOS หรือ DISK OPERATING SYSTEM) ซึ่งออกแบบให้มีลักษณะและการทำงาน ดังต่อไปนี้

3.8.2.1 ทำการติดต่อกับคำสั่ง

- โมดุลดอสรับการสั่งงานจากคำสั่ง
- โมดุลดอสทำหน้าที่เป็นทางเข้า (ENTRY POINT) สำหรับการสั่งให้ระบบทำงาน
- คำสั่งสั่งให้ระบบทำงาน โดยสั่งผ่านโมดุลดอส
- การทำงานของโมดุลดอส จะแบ่งออกเป็นฟังก์ชั่น (FUNCTION)
- คำสั่งสั่งให้ระบบทำงานโดยกำหนดหมายเลขฟังก์ชั่น
- หมายเลขฟังก์ชั่นของดอสเริ่มตั้งแต่ 0 ถึง 40
- การสั่งงานจะกำหนดหมายเลขฟังก์ชั่นไว้ที่รีจิสเตอร์ C และส่งข้อมูลผ่านรีจิสเตอร์ E หรือ DE
- เมื่อโมดุลดอสทำงานเสร็จ จะส่งผลลัพธ์ให้แก่คำสั่งผ่านรีจิสเตอร์ A หรือ HL

3.8.2.2 การส่งผ่านข้อมูลและผลลัพธ์

- กรณีที่ฟังก์ชั่นที่ต้องการ โมดุลดอสไม่สามารถทำได้ก็จะส่งต่อข้อมูลไปให้โมดุลอื่นทำงานแทน
- เมื่อโมดุลนั้นส่งผลลัพธ์กลับมาให้โมดุลดอส โมดุลดอสจะส่งผลลัพธ์กลับไปให้คำสั่งอีกต่อหนึ่ง

3.8.2.3 ทำการจัดการเกี่ยวกับแฟ้มข้อมูล

- ทำการจัดเก็บแฟ้มข้อมูล
- ทำการเก็บรายละเอียดเกี่ยวกับการใช้พื้นที่ผิวจานแม่เหล็ก

- ทำการอ่าน/บันทึกข้อมูลของแฟ้มกับจานแม่เหล็ก

การออกแบบโมดุลนี้ กำหนดให้การสั่งงานทำผ่านโมดุลดอสเท่านั้น ถึงแม้จะทราบว่า การทำงานนั้นโมดุลดอสไม่สามารถทำได้ก็ตาม แต่ก็ให้อาศัยโมดุลดอสเป็นทางผ่าน เหตุผลก็เพื่อให้การสั่งงานระบบ มีลักษณะที่ไม่ขึ้นอยู่กับเครื่องที่ใช้ (MACHINE INDEPENDENT) คำสั่งที่ใช้งานได้บนเครื่องหนึ่งสามารถนำมาใช้งานกับอีกเครื่องหนึ่งได้เช่นกัน แม้ว่าเครื่องทั้งสองจะมีขนาดหน่วยความจำและอุปกรณ์ที่ใช้ไม่เหมือนกันก็ตาม เพราะในการสั่งงานไม่จำเป็นต้องรู้ว่าฟังก์ชั่นนั้นอยู่ในโมดุลใด หรืออยู่ที่แอดเดรสใด แต่การสั่งงานทำโดยกำหนดหมายเลขฟังก์ชั่นที่ต้องการไว้ที่รีจิสเตอร์ C และส่งข้อมูลผ่านรีจิสเตอร์ E หรือ DE ซึ่งทำเหมือนกันทุกระบบ

3.8.3 <u>การออกแบบโมดุล III</u> โมดุลนี้จะตั้งชื่อเรียกว่า โมดุลไอโอ (I/O หรือ INPUT/OUTPUT) ซึ่งออกแบบให้มีลักษณะและการทำงานดังต่อไปนี้

3.8.3.1 ทำการติดต่อกับโมดุลดอส

- โมดุลไอโอประกอบด้วยหลายๆ รูทีนประกอบเข้าด้วยกัน
- แต่ละรูทีนมีการทำงานเฉพาะอย่าง
- โมดุลไอโอรับการสั่งงานจากโมดุลดอส
- การสั่งให้รูทีนใดในโมดุลไอโอทำงาน จะสั่งโดยเรียกผ่านแอดเดรสของรูทีนนั้น
- การส่งข้อมูลให้โมดุลไอโอส่งผ่านรีจิสเตอร์ B หรือ BC
- โมดุลไอโอส่งผลลัพธ์กลับผ่านรีจิสเตอร์ A หรือ HL

3.8.3.2 ทำการรับ/ส่งข้อมูลกับอุปกรณ์

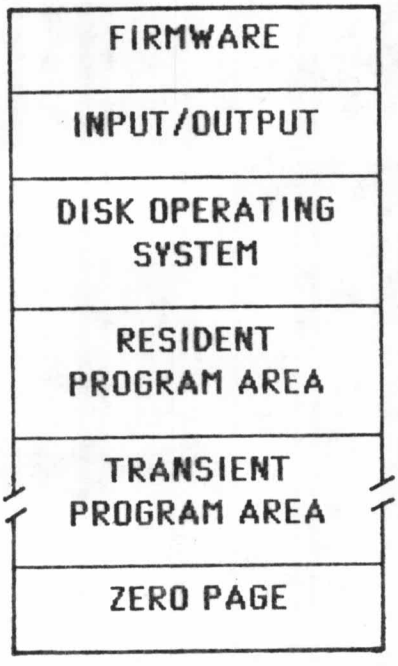
- แต่ละรูทีนของโมดุลไอโอ ทำหน้าที่รับ/ส่งข้อมูลกับอุปกรณ์แต่ละชนิด
- การเพิ่มอุปกรณ์ให้แก่ระบบ จะต้องเพิ่มรูทีนสำหรับอุปกรณ์นั้นในโมดุลไอโอด้วย

จากการออกแบบ ทำให้โมดุลไอโอมีลักษณะขึ้นอยู่กับเครื่องที่ใช้ (MACHINE DEPENDENT) เพราะแอดเดรสของแต่ละรูทีนของเครื่องหนึ่ง ย่อมไม่ตรงกันกับของอีกเครื่องหนึ่ง ถ้าเครื่องทั้งสองมีขนาดหน่วยความจำไม่เท่ากัน

แต่อย่างไรก็ตาม การที่แอดเดรสมีค่าไม่ตรงกันนี้ไม่มีผลกระทบต่อผู้ใช้ เพราะจากการออกแบบครอบคลุมโมดุลทั้งหมด กำหนดให้การสั่งงานระบบทำผ่านโมดุลดอสเสมอ ซึ่งสั่งตามหมายเลขฟังก์ชั่นของดอส โดยผู้ใช้ไม่ต้องสนใจว่ารูทีนของโมดุลไอโอจะอยู่ที่แอดเดรสใด

ในการตั้งชื่อโมดุลนี้ตั้งขึ้นตามลักษณะงานของโมดุล คือทำหน้าที่รับ/ส่ง (INPUT/OUTPUT) ข้อมูล

3.8.4 การออกแบบโมดุล IV โมดุลนี้เป็นผลจากการออกแบบฮาร์ดแวร์ของเครื่อง การทำงานของโมดุลขึ้นอยู่กับการทำงานของอุปกรณ์ที่ต่อ เช่น ต้องมีการควบคุมในการกำหนดเวลา(TIME OUT) และการหน่วงเวลา(DELAY LOOP) ในการติดต่อกับอุปกรณ์ โมดุลนี้ถูกเก็บเอาไว้ในรอม (ROM) ของเครื่อง ต่อเมื่อต้องการใช้งานจึงจะย้ายไปไว้ในส่วนที่เป็นแรม(RAM) โมดุลนี้จะตั้งชื่อเรียกว่าโมดุลเฟิร์มแวร์ (FIRMWARE)



รูปที่ 3.4 ตำแหน่งของโมดุลต่างๆ ในหน่วยความจำ

โมดุลอาร์พีเอ, โมดุลดอสและโมดุลไอโอจะถูกแปล (COMPILE) เป็นโปรแกรมภาษาเครื่อง (OBJECT PROGRAM) เก็บไว้ที่แทรค 0 และ แทรค 1 ของจานแม่เหล็ก ส่วนโมดุลเฟิร์มแวร์ถูกเก็บไว้ในรอม แต่เมื่อต้องการใช้งานจะทำการอ่านโมดุลทั้งหมดเข้ามาไว้ในหน่วยความจำ ดังรูปที่ 3.4 ดังนั้นในหน่วย

ความจำจะแบ่งพื้นที่ออกเป็น 2 ส่วน ส่วนหนึ่งสำหรับเก็บตัวโปรแกรมควบคุมระบบ และส่วนที่เหลือเป็นพื้นที่ชั่วคราว (TEMPORARY AREA) โดยมีพื้นที่ทีพีเอ (TPA หรือ TRANSIENT PROGRAM AREA) สำหรับเก็บโปรแกรมคำสั่งของผู้ใช้ และมีซีโร่เพจ (ZERO PAGE) สำหรับเก็บค่าชั่วคราวของระบบ

3.9 รายละเอียดของโมดุล

3.9.1 รายละเอียดของโมดุลอาร์พีเอ โมดุลอาร์พีเอประกอบด้วยโปรแกรมต่างๆ ดังต่อไปนี้

3.9.1.1 โปรแกรมทำหน้าที่รับข้อมูลจากแป้นพิมพ์ โปรแกรมส่วนนี้ทำหน้าที่รับบันทัดคำสั่งที่ผู้ใช้ป้อนผ่านทางแป้นพิมพ์

3.9.1.2 โปรแกรมทำหน้าที่เป็น LINE INTERPRETER โปรแกรมส่วนนี้ทำหน้าที่แปลความหมายของบันทัดคำสั่งที่ได้รับ เพื่อหาว่าผู้ใช้ต้องการใช้คำสั่งใด

3.9.1.3 โปรแกรมทำหน้าที่เป็นโหลดเดอร์ โปรแกรมส่วนนี้ทำหน้าที่อ่านคำสั่งจากจานแม่เหล็ก ลงไว้ในหน่วยความจำที่แอดเดรส 100H พร้อมกับกระโดดไปทำงานที่คำสั่งนั้น

3.9.1.4 โปรแกรมที่ทำหน้าที่เป็นคำสั่งประจำของระบบ โปรแกรมส่วนนี้ประกอบด้วย 9 โปรแกรมย่อย โดยแต่ละโปรแกรมย่อยทำหน้าที่เป็นแต่ละคำสั่งประจำของระบบ ได้แก่

- คำสั่งแสดงรายชื่อแฟ้มข้อมูล
- คำสั่งลบแฟ้มข้อมูล
- คำสั่งเปลี่ยนชื่อแฟ้มข้อมูล
- คำสั่งแสดงข้อมูลของแฟ้ม
- คำสั่งเก็บข้อมูลจากหน่วยความจำลงจานแม่เหล็ก
- คำสั่งกำหนดพาร์ติชั่น (PARTITION) ที่จะใช้งาน
- คำสั่งย้ายแฟ้มข้อมูลไปยังพาร์ติชั่นที่กำหนด
- คำสั่งแสดงสภาพของระบบ
- คำสั่งลบหน้าจอ

3.9.2 รายละเอียดของโมดุลดอส ในโมดุลดอสประกอบด้วยรูทีนย่อยๆ ซึ่งในการทำงานจะมีการใช้รูทีนร่วมกัน ดังนั้นการแยกการทำงานของโมดุลดอสจะแยกตามฟังก์ชั่นที่กระทำ ซึ่งได้แก่

3.9.2.1 ฟังก์ชั่นสร้างแฟ้มข้อมูล (CREATE FILE) ทำหน้าที่สร้างแฟ้มข้อมูลใหม่ขึ้นบนแผ่นจานแม่เหล็ก ขั้นตอนการทำงานแสดงในผังงานที่ 3.1

3.9.2.2 ฟังก์ชั่นเปิดแฟ้มข้อมูล (OPEN FILE) ทำหน้าที่เปิดแฟ้มข้อมูลที่สร้างไว้แล้ว ให้พร้อมสำหรับทำการอ่านหรือบันทึก ขั้นตอนการทำงานแสดงในผังงานที่ 3.2

3.9.2.3 ฟังก์ชั่นปิดแฟ้มข้อมูล (CLOSE FILE) ทำหน้าที่บันทึกข้อมูลที่กระทำครั้งหลังสุดลงเก็บบนจานแม่เหล็ก การปิดแฟ้มไม่จำเป็นต้องทำ ถ้าแฟ้มนั้นมีแต่การอ่านข้อมูลไปใช้ ขั้นตอนการทำงานแสดงในผังงานที่ 3.3

3.9.2.4 ฟังก์ชั่นลบแฟ้มข้อมูล (DELETE FILE) ทำหน้าที่ลบแฟ้มข้อมูลที่เก็บบนจานแม่เหล็กทิ้ง ขั้นตอนการทำงานแสดงในผังงานที่ 3.4

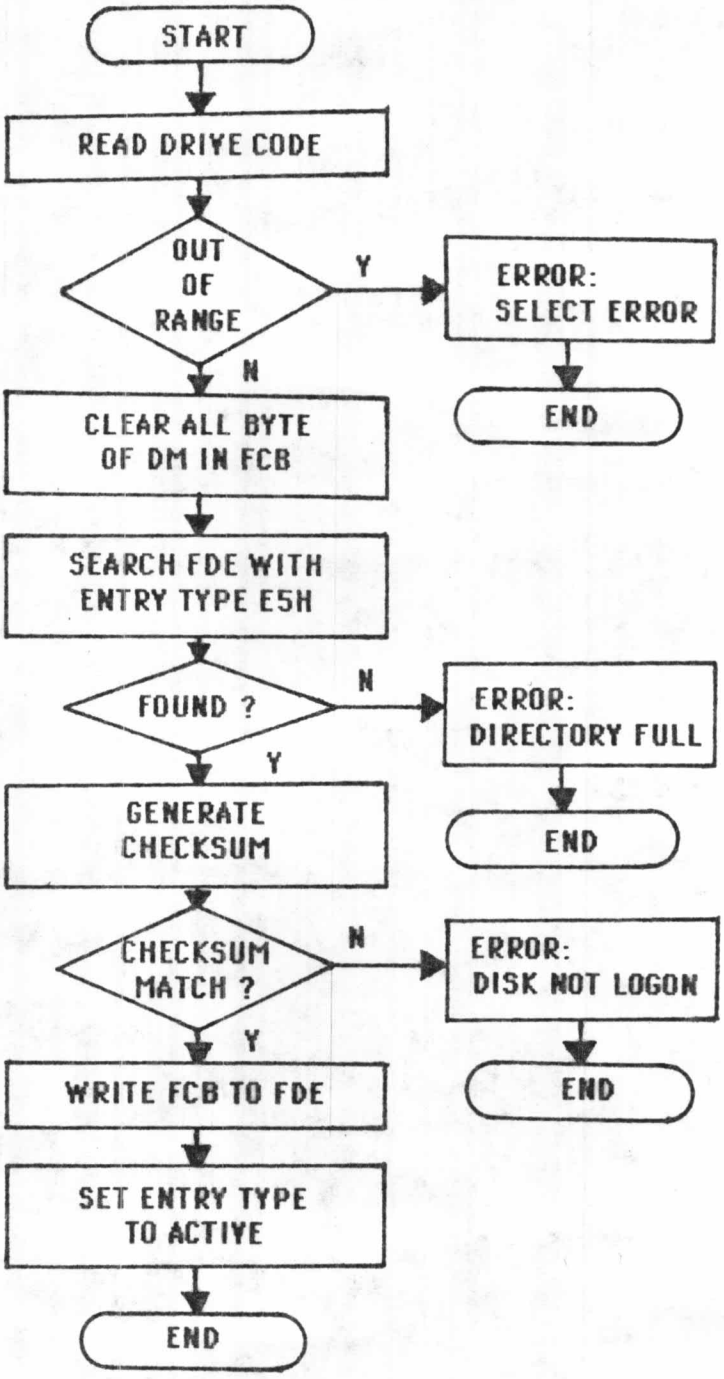
3.9.2.5 ฟังก์ชั่นอ่านแบบเรียงลำดับ (READ SEQUENTIAL) ทำหน้าที่อ่านข้อมูลของแฟ้มจากจานแม่เหล็ก เรียงต่อกันทีละระเบียน ขั้นตอนการทำงานแสดงในผังงานที่ 3.5

3.9.2.6 ฟังก์ชั่นบันทึกแบบเรียงลำดับ (WRITE SEQUENTIAL) ทำหน้าที่บันทึกข้อมูลลงบนจานแม่เหล็ก เรียงต่อกันทีละระเบียน ขั้นตอนการทำงานแสดงในผังงานที่ 3.6

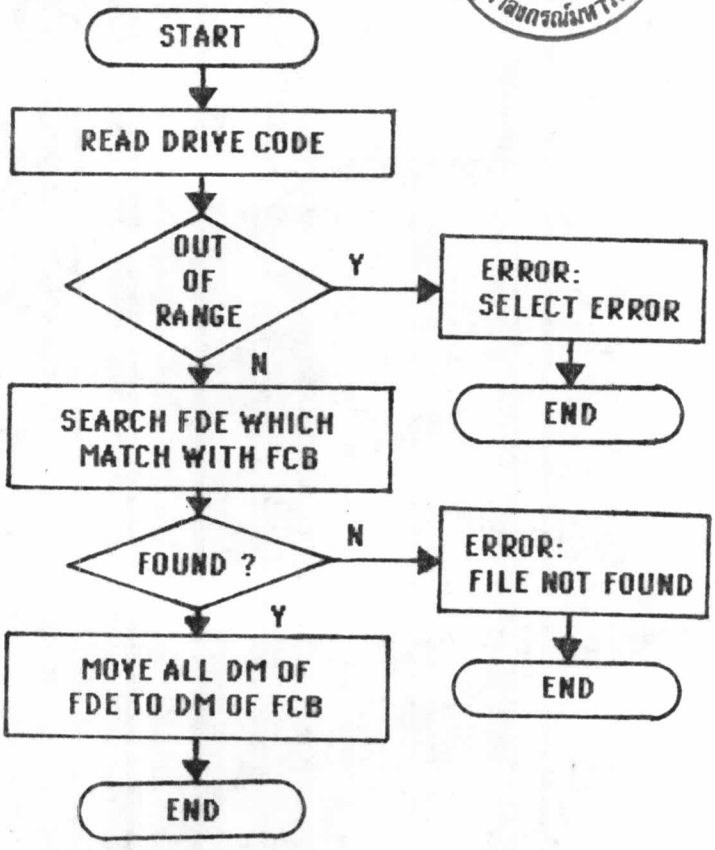
3.9.2.7 ฟังก์ชั่นอ่านแบบสุ่ม (READ RANDOM) ทำหน้าที่อ่านข้อมูลของแฟ้มเฉพาะระเบียนที่กำหนด โดยผู้ใช้เป็นผู้กำหนดหมายเลขระเบียนที่ต้องการอ่านเอาเอง

3.9.2.8 ฟังก์ชั่นบันทึกแบบสุ่ม (WRITE RANDOM) ทำหน้าที่บันทึกข้อมูลลงจานแม่เหล็กตรงตำแหน่งระเบียนที่กำหนด โดยผู้ใช้เป็นผู้กำหนดหมายเลขระเบียนที่จะเก็บข้อมูลเอาเอง

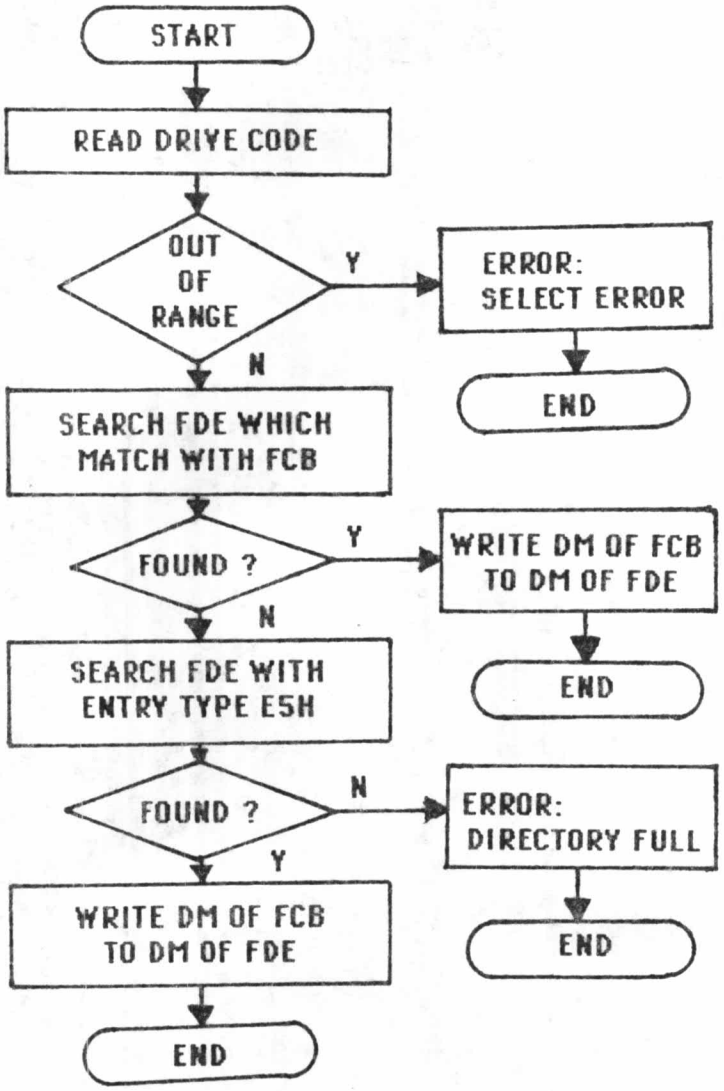
3.9.2.9 ฟังก์ชั่นแปลงหมายเลขระเบียนแบบสุ่ม (CONVERT RANDOM RECORD) ทำหน้าที่แปลงหมายเลขระเบียนแบบสุ่มที่ผู้ใช้กำหนด ให้อยู่ในรูปแบบเดียวกับหมายเลขระเบียนแบบเรียงลำดับ เพื่อทำการอ่าน/บันทึกข้อมูลที่ระเบียนนั้นต่อไป ขั้นตอนการทำงานแสดงในผังงานที่ 3.7



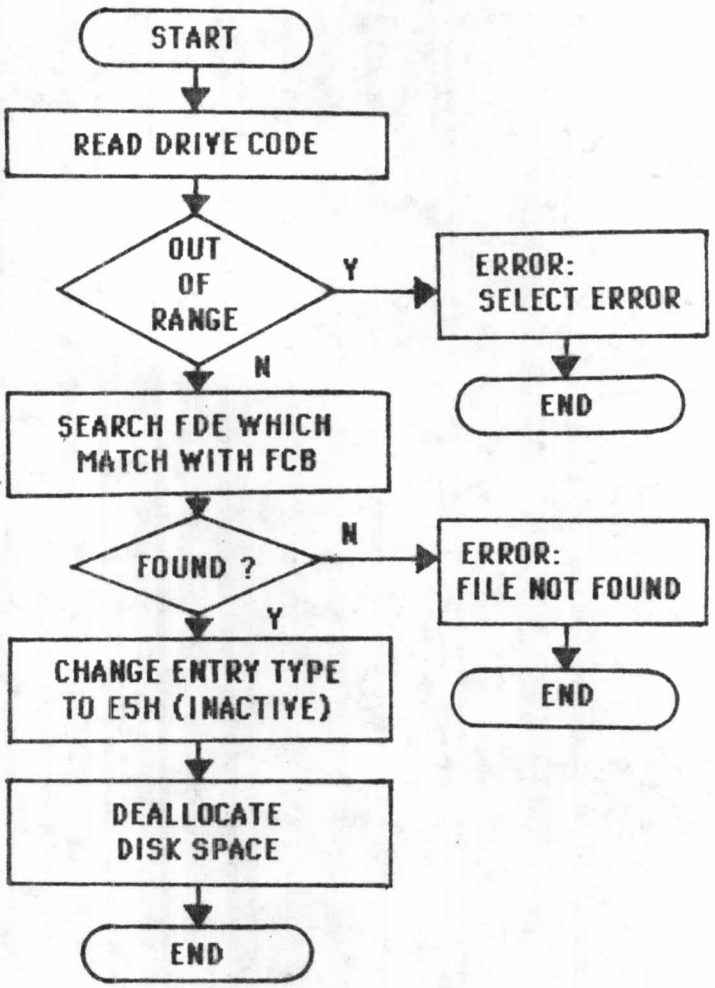
ผังงานที่ 3.1 การสร้างแฟ้มข้อมูล



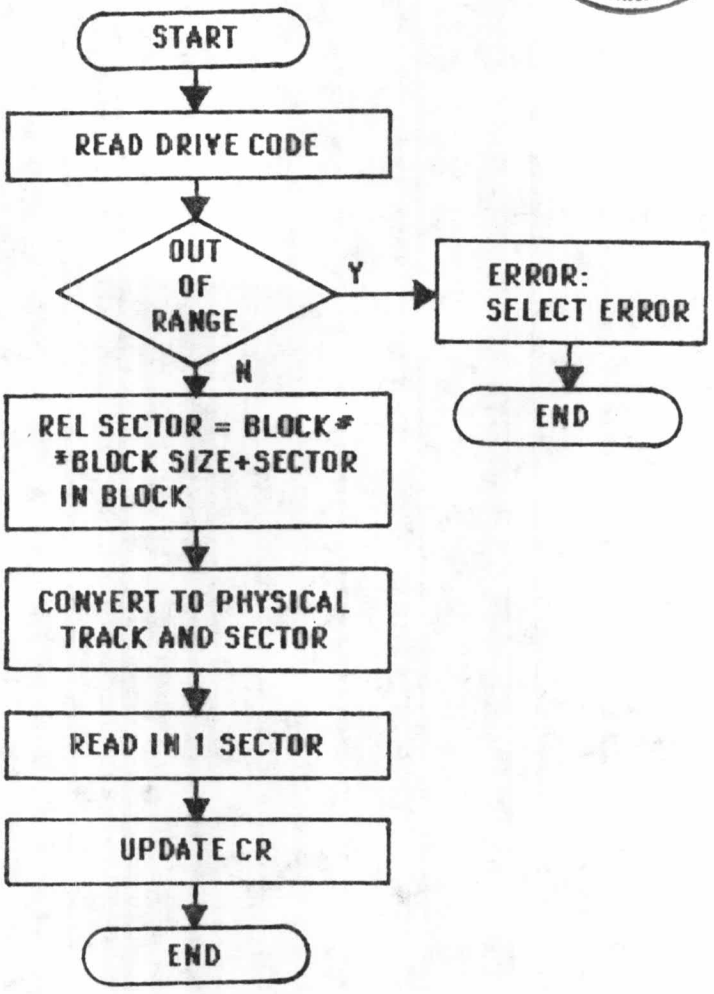
ผังงานที่ 3.2 การเปิดแฟ้มข้อมูล



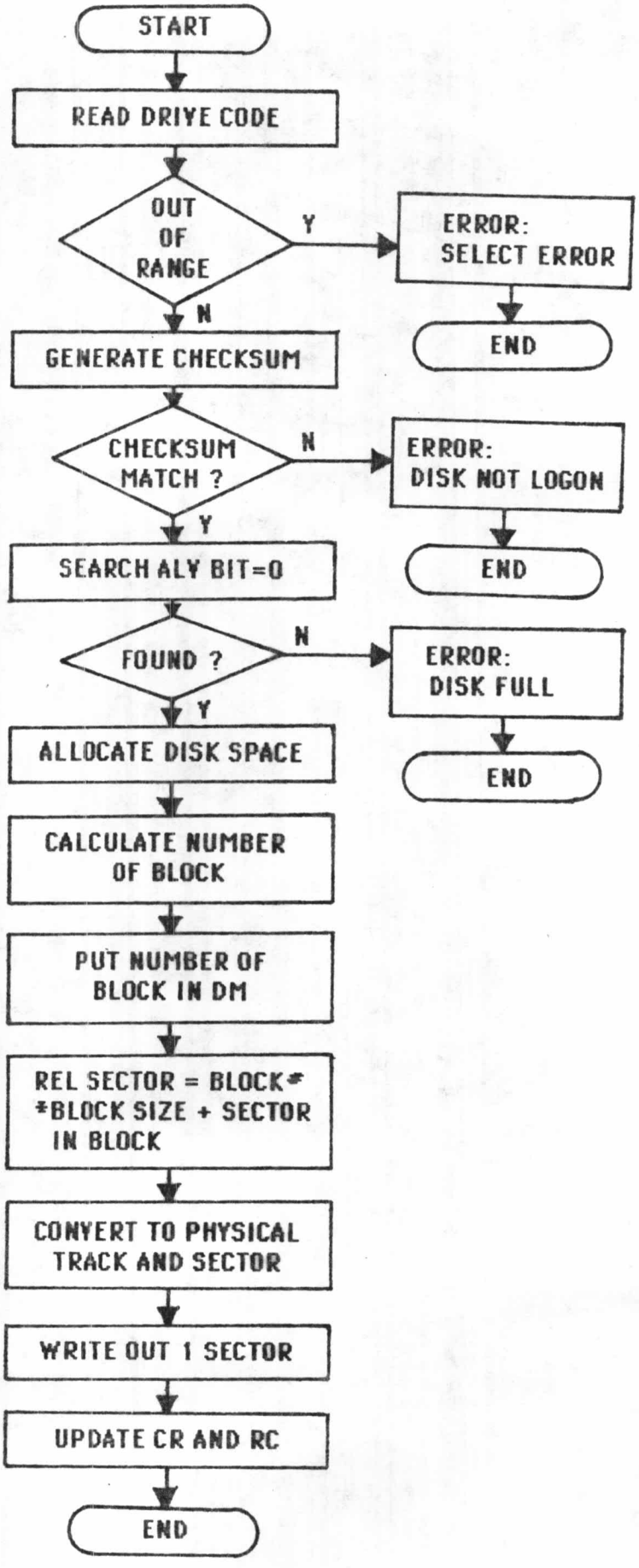
ผังงานที่ 3.3 การปิดแฟ้มข้อมูล



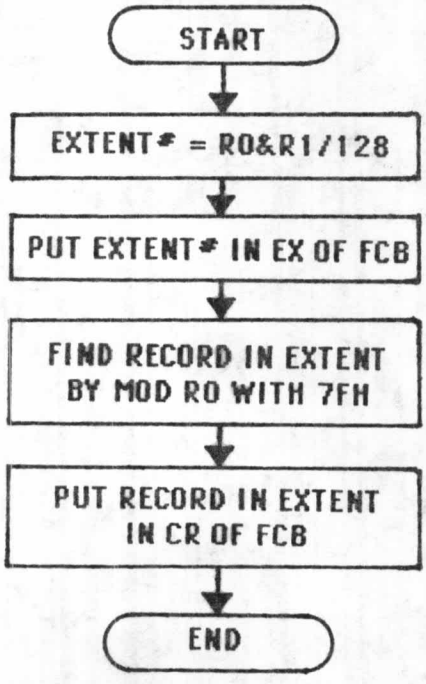
ผังงานที่ 3.4 การลบแฟ้มข้อมูล



ผังงานที่ 3.5 การอ่านแบบเรียงลำดับ



ผังงานที่ 3.6 การบันทึกแบบเรียงลำดับ



ผังงานที่ 3.7 การแปลงหมายเลขระเบียนแบบสุ่ม

3.9.3 รายละเอียดของโมดุลไอโอ โมดุลไอโอประกอบด้วยรูทีนสำหรับทำหน้าที่ต่างๆ 17 รูทีน ดังต่อไปนี้

3.9.3.1 รูทีน "CBOOT" ทำหน้าที่กำหนดค่าเริ่มต้นให้แก่ระบบเพื่อให้ระบบพร้อมที่จะทำงาน รูทีนนี้จะใช้เพียงครั้งเดียวตอนทำการเปิดเครื่อง

3.9.3.2 รูทีน "WBOOT" ทำหน้าที่อ่านโปรแกรมควบคุมระบบจากจานแม่เหล็กเข้ามาไว้ในหน่วยความจำ รูทีนนี้จะทำเมื่อต้องการส่งการควบคุมคืนให้แก่โปรแกรมควบคุมระบบ

3.9.3.3 รูทีน "CONST" ทำหน้าที่ตรวจสภาวะของคอนโซลว่ามีการส่งข้อมูลเข้ามาหรือไม่

3.9.3.4 รูทีน "CONIN" ทำหน้าที่รับข้อมูลจากคอนโซล เข้ามาเก็บไว้ในรีจิสเตอร์ A

3.9.3.5 รูทีน "CONOUT" ทำหน้าที่ส่งข้อมูลจากรีจิสเตอร์ C ออกทางคอนโซล

3.9.3.6 รูทีน "LIST" ทำหน้าที่ส่งข้อมูลจากรีจิสเตอร์ C ออกทางอุปกรณ์ LIST ซึ่งปรกติจะกำหนดให้เป็นเครื่องพิมพ์

3.9.3.7 รูทีน "LISTST" ทำหน้าที่ตรวจสภาวะของอุปกรณ์ LIST ว่าพร้อมจะรับข้อมูลหรือไม่

3.9.3.8 รูทีน "PUNCH" ทำหน้าที่ส่งข้อมูลจากรีจิสเตอร์ C ออกทางเครื่องเจาะเทปกระดาษ

3.9.3.9 รูทีน "READER" ทำหน้าที่อ่านข้อมูลจากเครื่องอ่านเทปกระดาษ เข้ามาเก็บไว้ในรีจิสเตอร์ A

3.9.3.10 รูทีน "HOME" ทำหน้าที่เลื่อนหัวอ่านของเครื่องขับจานแม่เหล็กไปยังแทรค 0

3.9.3.11 รูทีน "SETDRV" ทำหน้าที่เลือกเครื่องขับจานแม่เหล็ก

3.9.3.12 รูทีน "SETTRK" ทำหน้าที่กำหนดแทรค

3.9.3.13 รูทีน "SETSEC" ทำหน้าที่กำหนดเซกเตอร์

3.9.3.14 รูทีน "SETDMA" ทำหน้าที่กำหนด DMA แอดเดรส

3.9.3.15 รูทีน "READ" ทำหน้าที่อ่านข้อมูลจากจานแม่เหล็กที่แทรคและเซกเตอร์ตามที่กำหนด ไปไว้ที่ DMA แอดเดรสที่กำหนด

3.9.3.16 รูทีน "WRITE" ทำหน้าที่บันทึกข้อมูลจาก DMA แอด-

เดรสที่กำหนด ไปเก็บบนจานแม่เหล็กที่แทรคและเซกเตอร์ที่กำหนด

3.9.3.17 รูทีน "SECTRN" ทำหน้าที่แปลงหมายเลขเซกเตอร์ของดอสให้เป็นหมายเลขเซกเตอร์จริงบนจานแม่เหล็ก

3.9.4 รายละเอียดของโมดุลเพิร์มแวร์ โมดุลเพิร์มแวร์ประกอบด้วยรูทีนต่างๆ ซึ่งแต่ละรูทีนใช้สำหรับทำการติดต่อกับอุปกรณ์แต่ละชนิด ดังนั้นจำนวนรูทีนในโมดุลนี้อาจเพิ่มขึ้นได้ ถ้าทำการเพิ่มเติมอุปกรณ์ให้แก่ระบบ สำหรับระบบที่ใช้อยู่ โมดุลเพิร์มแวร์ประกอบด้วยรูทีน 14 รูทีน ดังต่อไปนี้

3.9.4.1 รูทีน "TMSTAT" ทำหน้าที่ตรวจสอบสภาวะของแป้นพิมพ์ โดยถ้ามีตัวอักษรส่งเข้ามาซีโร่แฟลก (ZERO FLAG) จะเป็น 1 แต่ถ้าไม่มีการส่งซีโร่แฟลกจะเป็น 0

3.9.4.2 รูทีน "CIN" ทำหน้าที่รับข้อมูลจากแป้นพิมพ์ โดยทำการรอรับข้อมูลที่ส่งเข้ามาจากแป้นพิมพ์ นำมาเก็บไว้ที่รีจิสเตอร์ A

3.9.4.3 รูทีน "COUT" ทำหน้าที่ส่งข้อมูลออกจอภาพ โดยเมื่อตรวจพบว่าจอภาพพร้อมที่จะรับข้อมูล จะส่งข้อมูลจากรีจิสเตอร์ A ออกไปให้

3.9.4.4 รูทีน "PRTSTA" ทำหน้าที่ตรวจดูว่าเครื่องพิมพ์พร้อมจะรับข้อมูลหรือไม่ ถ้าพร้อมซีโร่แฟลกจะเป็น 1 แต่ถ้าไม่พร้อมซีโร่แฟลกจะเป็น 0

3.9.4.5 รูทีน "PRTOUT" ทำหน้าที่ส่งข้อมูลออกทางเครื่องพิมพ์ โดยเมื่อเครื่องพิมพ์พร้อมที่จะรับข้อมูล จะส่งข้อมูลจากรีจิสเตอร์ A ออกไปให้

3.9.4.6 รูทีน "HOME" ทำหน้าที่เลื่อนหัวอ่านของเครื่องขับจานแม่เหล็กไปอยู่ที่แทรค 0

3.9.4.7 รูทีน "SEEK" ทำหน้าที่เลื่อนหัวอ่านของเครื่องขับจานแม่เหล็ก ไปยังแทรคที่กำหนดไว้ในรีจิสเตอร์ C

3.9.4.8 รูทีน "SECSET" ทำหน้าที่กำหนดเซกเตอร์ที่จะทำการอ่าน/บันทึก ตามค่าที่กำหนดไว้ในรีจิสเตอร์ C

3.9.4.9 รูทีน "DENFIX" ทำหน้าที่กำหนดขนาดของเซกเตอร์ที่จะใช้ในการอ่าน/บันทึก

3.9.4.10 รูทีน "SIDEFX" ทำหน้าที่กำหนดด้านของแผ่นจานแม่เหล็ก

3.9.4.11 รูทีน "DRIVE" ทำหน้าที่เลือกเครื่องขับจานแม่เหล็กที่จะใช้ในการอ่าน/บันทึก

3.9.4.12 รูทีน "DMA" ทำหน้าที่กำหนด DMA แอดเดรสในหน่วยความจำ ตามค่าที่กำหนดไว้ในรีจิสเตอร์ BC เพื่อใช้ในการอ่าน/บันทึกระหว่างหน่วยความจำกับจานแม่เหล็ก

3.9.4.13 รูทีน "READ" ทำหน้าที่อ่านข้อมูลจากจานแม่เหล็กมาเก็บไว้ที่ DMA แอดเดรส

3.9.4.14 รูทีน "WRITE" ทำหน้าที่บันทึกข้อมูลจาก DMA แอดเดรสลงเก็บบนแผ่นจานแม่เหล็ก

3.10 วิธีการเขียนโปรแกรม

ในการเขียนโปรแกรมแต่ละโมดุล เพื่อนำมาประกอบเข้าด้วยกันเป็นโปรแกรมควบคุมระบบ ไม่สามารถเขียนโดยออกแบบขั้นตอนการทำงานของโปรแกรมขึ้นมาใหม่ได้ทั้งหมด เพราะต้องคำนึงถึงข้อจำกัดอันหนึ่งซึ่งวางไว้ คือโปรแกรมควบคุมระบบที่สร้างขึ้น ต้องสามารถใช้งานกับโปรแกรมสำเร็จรูปที่มีอยู่ได้ นั่นคือ บางขั้นตอนของโปรแกรมควบคุมระบบ ขึ้นอยู่กับมาตรฐานที่โปรแกรมสำเร็จรูปใช้ในการติดต่อ, สั่งงาน และรับ/ส่งข้อมูลกับระบบ

การที่จะรู้ถึงมาตรฐานที่ใช้ในการติดต่อ ไม่สามารถหาจากโปรแกรมสำเร็จรูปแต่ละโปรแกรมได้โดยตรง เพราะโปรแกรมสำเร็จรูปที่ใช้อยู่มีหลายโปรแกรม การจะหามาตรฐานของแต่ละโปรแกรมสำเร็จรูป เป็นงานที่ยากและเสียเวลามาก ผู้ทำวิจัยจึงใช้หลักว่า โปรแกรมสำเร็จรูปทั้งหลายสามารถทำงานภายใต้ระบบ CP/M ดังนั้น CP/M จึงต้องมีมาตรฐานที่ใช้สำหรับติดต่อกับโปรแกรมสำเร็จรูปเหล่านี้อยู่ในตัวมันเอง ถ้ารู้ถึงมาตรฐานของ CP/M ก็เท่ากับรู้มาตรฐานที่ใช้ในการติดต่อกับโปรแกรมสำเร็จรูปทั้งหมด

การเขียนโปรแกรมแต่ละโมดุลขึ้นมา เริ่มแรกจะหามาตรฐานในการติดต่อของ CP/M ก่อน โดยทำการดัมพ์ (DUMP) โมดุลของ CP/M จากหน่วยความจำซึ่งอยู่ในรูปของโปรแกรมภาษาเครื่อง (OBJECT PROGRAM) มาทำการแปลงกลับเป็นโปรแกรมแอสเซมบลี้ จากนั้นจึงศึกษาโปรแกรมแอสเซมบลี้นั้นเพื่อหามาตรฐานที่ CP/M ใช้ เมื่อรู้มาตรฐานนั้นแล้วจึงทำการเขียนโปรแกรมขึ้นมาใหม่ด้วยชุดคำสั่ง (INSTRUCTION SET) ของ Z80 ให้มีมาตรฐานตรงกับที่หามาได้ ดังนั้น แต่ละโมดุลที่สร้างขึ้น จะมีบางส่วนที่มีขั้นตอนการทำงานเหมือนกับ

ของ CP/M ซึ่งเป็นสิ่งที่ไม่สามารถหลีกเลี่ยงได้ เพื่อให้ยังคงสามารถใช้โปรแกรมสำเร็จรูปที่มีอยู่ได้

อย่างไรก็ตาม เนื่องจากโปรแกรมที่เขียนขึ้นใช้ชุดคำสั่งของ Z80 ซึ่งมีขนาดของคำสั่ง (INSTRUCTION LENGTH) สั้นกว่าชุดคำสั่งของ 8080 ซึ่งใช้ใน CP/M ทำให้โปรแกรมควบคุมระบบที่สร้างขึ้นมีความสามารถเหนือกว่า CP/M เพราะเมื่อทำการเขียนโปรแกรมครอบคลุมการทำงานครบตามที่ CP/M สามารถทำได้แล้ว ยังมีเนื้อที่หน่วยความจำเหลืออยู่อีก ซึ่งเนื้อที่ส่วนนี้เป็นส่วนที่ใช้เก็บโปรแกรมสำหรับการทำงานนอกเหนือไปจากความสามารถที่ CP/M จะทำได้ ตัวอย่างได้แก่

ในโมดุลอาร์พีเอ

- ขยายความยาวของชื่อคำสั่งประจำจากสูงสุด 4 ตัวอักษร เป็น 8 ตัวอักษร
- เพิ่มคำสั่งประจำจาก 6 คำสั่งเป็น 9 คำสั่ง
- เพิ่มโปรแกรมแสดงคำแนะนำสำหรับคำสั่งประจำทั้งหมด
- สร้างตารางเก็บชื่อคำสั่งประจำขึ้นใหม่ เพื่อให้สามารถเปลี่ยนชื่อของคำสั่งประจำได้
- ขยายความยาวของพร็อมจาก 1 ตัวอักษร เป็น 8 ตัวอักษร
- สร้างบัฟเฟอร์เก็บพร็อมและเครื่องหมายพร็อม เพื่อให้สามารถเปลี่ยนพร็อมและเครื่องหมายพร็อมได้
- เปลี่ยนข้อความแสดงความผิดพลาดให้มีความละเอียดขึ้น สามารถบอกได้ถึงระดับคำสั่งหรือแฟ้มข้อมูลที่ผิดพลาด ซึ่ง CP/M ไม่สามารถทำได้

ในโมดุลดอส

- สร้างฟังก์ชั่นที่ 38 ขึ้นมาใหม่ เพื่อสนับสนุนคำสั่งประจำที่สร้างเพิ่มขึ้นในโมดุลอาร์พีเอ
- เปลี่ยนข้อความแสดงความผิดพลาดให้มีความละเอียดขึ้น และสื่อความหมายมากขึ้น

ในโมดุลไอโอ

- สร้างโปรแกรมสำหรับแสดงภาพโลโก เมื่อทำการเปิดเครื่อง
- สร้างบัฟเฟอร์สำหรับเก็บคำสั่งออโตรัน ทำให้สามารถเลือกกำหนดคำสั่งใดๆ เป็นคำสั่งออโตรันได้

ในภาคผนวก จ,ฉ และ ช เป็นตัวโปรแกรมของแต่ละโมดุล โปรแกรมส่วนที่ทำงานเพิ่มเติมจากการทำงานของ CP/M จะมีเส้นล้อมกรอบไว้.