บทที่ ๕

สรุปผลการวิจัย

จากการศึกษาลักษณะการจัดแฟ้มข้อมูล การเรียกใช้แฟ้มข้อมูล ตลอดจนภาษาที่ใช้กับแฟ้มข้อมูลซึ่งผู้วิจัยได้ใช้คือ ภาษาโคบอล ภาษาฟอร์แทรนและภาษาแอสเซมเบอร์ และจากการออกแบบปฏิบัติจริงทำให้ผู้วิจัยพอสรุปผลการวิจัยได้โดย

- เปรียบเทียบข้อดีข้อเสียของการจัดแฟ้มข้อมูลแต่ละวิธี
- เปรียบเทียบภาษาที่ใช้กับวิธีการจัดแฟ้มข้อมูลแบบต่าง ๆ

๕.๑ เปรียบเทียบข้อดีข้อเสียของการจัดแฟ้มข้อมูลแต่ละวิธี

๕.๑.๑ การจัดแฟ้มข้อมูลแบบแซม

ข้อดี	ข้อเสีย
๑. ใช้เนื้อที่ในจานแม่เหล็กน้อยที่สุดเมื่อเปรียบเทียบกับวิธีการอื่น ๆ เพราะใช้เนื้อที่เฉพาะเก็บข้อมูลหลักเท่านั้น	๑. เสียเวลาในการค้นหาระเบียนข้อมูลที่ต้องการนาน
๒. ระเบียนข้อมูลที่ใช้จะมีลักษณะของรูปแบบอย่างไรก็ได้คือมีความยาวคงที่ ไม่คงที่ หรือไม่กำหนดความยาว	๒. ในการเปลี่ยนแปลงแก้ไขแฟ้มข้อมูลจะต้องมีแฟ้มข้อมูลใหม่และทุกระเบียนข้อมูลของแฟ้มข้อมูลหลักเก่าและใหม่จะถูกอ่านและบันทึก
๓. สามารถใช้แฟ้มข้อมูลหลักเดิมเป็นข้อมูลสำรองได้โดยอัตโนมัติ	

๕.๑.๒ การจัดแฟ้มข้อมูลแบบแคม

ข้อดี	ข้อเสีย
๑. สามารถเข้าถึงระเบียนข้อมูลที่ต้องการได้โดยตรง ทำให้เสียเวลาน้อย	๑. ผู้ใช้ต้องรับผิดชอบในการที่จัดการกับระเบียนข้อมูลส่วนเกิน การเพิ่มระเบียนข้อมูลลงในแฟ้มข้อมูล
๒. ใช้เนื้อที่ในจานแม่เหล็กน้อยกว่าแบบไอแซม	๒. ในกรณีที่คำนวณแล้วได้ค่าตำแหน่งที่เก็บซ้ำกันมาก ๆ จะทำให้เสียเวลาในการค้นหามาก
๓. ในการเปลี่ยนแปลงแก้ไขระเบียนข้อมูลไม่จำเป็นต้องมีแฟ้มข้อมูลหลักใหม่ และระเบียนข้อมูลในแฟ้มข้อมูลเปลี่ยนแปลงไม่ต้องเรียงลำดับเช่นเดียวกับในแฟ้มข้อมูลหลัก	๓. ไม่สามารถประมวลผลแบบเรียงลำดับได้
	๔. ขาดระบบข้อมูลสำรองโดยอัตโนมัติ

๕.๑.๓ การจัดแฟ้มข้อมูลแบบไอแซม

ข้อดี	ข้อเสีย
๑. ประมวลผลได้ทั้งแบบเรียงลำดับและเข้าถึงข้อมูลได้โดยตรง	๑. ต้องเตรียมที่ว่างเผื่อไว้สำหรับการเพิ่มระเบียนข้อมูลลงในแฟ้มข้อมูล
๒. ในการเปลี่ยนแปลงแก้ไขระเบียนข้อมูลไม่จำเป็นต้องมีแฟ้มข้อมูลหลักใหม่ และระเบียนข้อมูลในแฟ้มข้อมูลเปลี่ยนแปลงไม่ต้องเรียงลำดับเช่นเดียวกับในแฟ้มข้อมูลหลัก	๒. ต้องเสียเนื้อที่สำหรับส่วนที่เป็นดัชนี
๓. ลดภาระของผู้เขียนโปรแกรมคือไม่ต้องเขียนส่วนที่เป็นการจัดการข้อมูล เพราะไอแซมมี Access Method Mechanism ที่ช่วยในการจัดการข้อมูลและสอดแทรกอยู่แล้ว	๓. กรณีที่มีการลบ หรือเพิ่มระเบียนข้อมูลมาก ๆ จำเป็นต้องมีการจัดแฟ้มข้อมูลใหม่
	๔. ขาดระบบข้อมูลสำรองโดยอัตโนมัติ

๕.๑.๔ การจัดแฟ้มข้อมูลแบบวีแซม

ข้อดี	ข้อเสีย
๑. เพิ่มประสิทธิภาพในการทำงาน เพิ่มความแม่นยำและความปลอดภัยของข้อมูล	๑. ผู้ใช้ต้องเสียเวลาในการ Define Master Catalog, Space, Cluster
๒. สามารถประมวลผลแฟ้มข้อมูลได้ทั้งแบบเรียงลำดับและเข้าถึงแฟ้มข้อมูลได้โดยตรง	๒. กรณีมีข้อผิดพลาดเกิดขึ้นก็ไม่สามารถเปิด (open) แฟ้มข้อมูลได้ต้องทำการตรวจสอบ (Verify) แฟ้มข้อมูลก่อน
๓. ลดภาระของผู้เขียนโปรแกรม คือไม่ต้องเขียนส่วนที่เป็นการจัดการข้อมูล เพราะวีแซมมี Access Method Mechamism ที่ช่วยในการจัดการข้อมูลและสอดแทรกอยู่แล้ว	๓. ขาดระบบสำรองข้อมูลโดยอัตโนมัติ

๕.๒ เปรียบเทียบภาษาที่ใช้กับวิธีการจัดแฟ้มข้อมูลต่าง ๆ

ในการเปรียบเทียบภาษากับวิธีการจัดแฟ้มข้อมูลแบบต่าง ๆ นี้ จะพิจารณาแยกตามวิธีการจัดแฟ้มข้อมูลวิธีต่าง ๆ ดังนี้

๕.๒.๑ เปรียบเทียบภาษาฟอร์แทรน โคบอล และแอสเซมเบอร์กับการจัดแฟ้มข้อมูลแบบแซม

๕.๒.๒ เปรียบเทียบภาษาฟอร์แทรน โคบอล และแอสเซมเบอร์กับการจัดแฟ้มข้อมูลแบบแดม

๕.๒.๓ เปรียบเทียบภาษาโคบอล และแอสเซมเบอร์กับการจัดแฟ้มข้อมูลแบบไอแซม

๕.๒.๔ เปรียบเทียบภาษาโคบอล และแอสเซมเบอร์กับการจัดแฟ้มข้อมูลแบบวีแซม

สำหรับการเปรียบเทียบแต่ละวิธีดังกล่าวข้างต้นนี้ ผู้วิจัยจะแสดงการเปรียบเทียบในรูปแบบของตารางโดยพิจารณาถึงหัวข้อต่าง ๆ ดังต่อไปนี้

- ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถ
- ลักษณะของการใช้โปรแกรม

- Compile and Execution Time
- File Protection and Error Message

๕.๒.๑ เปรียบเทียบภาษาฟอร์แทรน โคบอล และแอสเซมเบอร์กับการจัดแฟ้มข้อมูลแบบแซม (SAM)

๕.๒.๑.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถ

ในการพิจารณาถึงลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถนี้ จะแยกพิจารณาดังนี้

ก. ลักษณะแฟ้มข้อมูล

ข. ลักษณะระเบียนข้อมูล

ค. เนื้อที่ที่ใช้ในการเก็บระเบียนข้อมูล

จากตารางที่ ๕.๒.๑.๑ จะเห็นได้ว่าภาษาโคบอล และแอสเซมเบอร์มีความคล่องตัวในการจัดแฟ้มข้อมูลมากกว่า ภาษาฟอร์แทรน ดังจะเห็นได้ว่า ระเบียนข้อมูลที่ใช้กับภาษาโคบอลและแอสเซมเบอร์นั้น สามารถจัดให้อยู่ในลักษณะใดก็ได้ กล่าวคือ ระเบียนข้อมูลอาจมีความยาวคงที่ไม่คงที่ หรือไม่กำหนดความยาวก็ได้ และระเบียนข้อมูลอาจถูกจัดให้เป็นบล็อคหรือไม่เป็นบล็อคก็ได้ ในขณะที่ภาษาฟอร์แทรนนั้น ความยาวของระเบียนข้อมูลจะต้องคงที่ และไม่สามารถจัดให้เป็นบล็อคได้จึงทำให้เสียเนื้อที่ในการจัดเก็บระเบียนข้อมูลมากกว่าเมื่อเทียบกับการจัดแฟ้มข้อมูลโดยใช้ภาษาโคบอลและแอสเซมเบอร์

ตารางที่ ๕.๒.๑.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถในการจัดแฟ้มข้อมูลแบบแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑. ลักษณะแฟ้มข้อมูล	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๑.๑	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๑.๑	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๑.๑
๒. ลักษณะระเบียนข้อมูล	ก. ความยาวของระเบียนข้อมูลต้องคงที่ ข. ไม่สามารถจัดให้เป็นบล็อคได้	ก. ระเบียนข้อมูลอาจมีความยาวคงที่ไม่คงที่หรือไม่กำหนดความยาวก็ได้ ข. อาจจัดให้เป็นบล็อคหรือไม่เป็นบล็อคก็ได้	ก. ลักษณะเช่นเดียวกับโคบอล ข. เช่นเดียวกับโคบอล
๓. เนื้อที่ที่ใช้ในการเก็บระเบียนข้อมูล	ใช้เนื้อที่ในการจัดเก็บระเบียนข้อมูลมาก เพราะไม่สามารถจัดระเบียนข้อมูลเป็นแบบบล็อคได้ ทำให้เสียเนื้อที่ระหว่างระเบียนข้อมูล (Interblock gap) มาก	ในกรณีจัดระเบียนข้อมูลเป็นบล็อค จะใช้เนื้อที่ในการเก็บระเบียนข้อมูลน้อยกว่าแบบไม่จัดเป็นบล็อค เนื่องจากสามารถลดจำนวนเนื้อที่ระหว่างระเบียนข้อมูลได้	เช่นเดียวกับโคบอล

๕.๒.๑.๒ ลักษณะของการใช้โปรแกรม

ในการพิจารณานี้จะแยกพิจารณาตามหัวข้อดังนี้คือ

ก. การเขียนโปรแกรม

ข. คำสั่ง Read

ค. คำสั่ง Write

ง. คำสั่ง Close

จ. คำสั่ง OPEN

จากตารางที่ ๕.๒.๑.๒ จะเห็นว่าการใช้โปรแกรมของภาษาฟอร์แทรน ใช้ง่ายและสะดวกกว่าภาษาโคบอลและแอสเซมเบอร์ เนื่องจากว่า ผู้ใช้ไม่ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูล รวมทั้งรายละเอียดต่าง ๆ ของระเบียนข้อมูลไว้ก่อน เมื่อจะใช้ก็อ้างถึงได้เลยในรูปของ Format แต่ภาษาฟอร์แทรนก็มีข้อเสียคือ ไม่มีสัญญลักษณ์ที่จะกำหนดรูปแบบรายการย่อยของข้อมูล (ตัวเลข) ที่จะพิมพ์ออกมาให้อยู่ในรูปแบบที่อ่านง่าย สวยงาม และมีความหมายในการบัญชีได้เช่นเดียวกับภาษาโคบอลและแอสเซมเบอร์ ดังนั้น ภาษาฟอร์แทรนจึงไม่เหมาะกับงานที่จะต้องออกรายงานมาก ๆ เหมาะสำหรับงานที่ใช้ในการคำนวณมากกว่า ถ้างานที่ต้องออกรายงานมาก ๆ ควรใช้ภาษาโคบอล สำหรับแอสเซมเบอร์นั้นแม้ว่าสามารถทำได้เช่นเดียวกับภาษาโคบอล แต่ก็มีความยุ่งยากในการใช้มากกว่า และแอสเซมเบอร์ของคอมพิวเตอร์เครื่องใดเครื่องหนึ่งนั้นก็ไม่สามารถนำไปใช้กับเครื่องอื่น ๆ ได้

ตารางที่ ๕.๒.๑.๒ ลักษณะของการใช้โปรแกรมในการจัดแฟ้มข้อมูลแบบแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑. การเขียนโปรแกรม	๑. สามารถเขียนโปรแกรมได้สั้น (ภาคผนวก ง) ๒. ผู้ใช้ไม่ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลไว้ก่อน	๑. เขียนโปรแกรมยาว (ภาคผนวก ง) ๒. ผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลใน ก. INPUT-OUTPUT SECTION โดยกำหนดที่ SELECT ซึ่งมีรูปแบบดังนี้ SELECT [OPTIONAL] file-name ASSIGN To SYSnnn-class-device-organization [-name] -filename เป็นชื่อข้อมูลที่นักเขียนโปรแกรมใช้ -nnn ใช้แทนชื่อของแฟ้มข้อมูล เป็นตัวเลข ๓ หลัก มีค่าจาก ooo ถึง ๒๒๑ -class ทำหน้าที่บอกถึงประเภทเครื่องอ่าน/บันทึกข้อมูล -device ทำหน้าที่บอกหมายเลขของเครื่องอ่านหรือบันทึกข้อมูลแต่ละชนิดแยกตาม class	๑. ผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูล (ภาคผนวก ง) โดยกำหนดที่ DTF macro ซึ่งมีรูปแบบดังนี้ filename DTFSD BLKSIZE = nnnn DEVADDR = SYSnnn DEVICE = nnnn IOAREA1 = xxxxxxxx RECFORM = xxxxxx RECSIZE = nnnnn TYPEFLE = xxxxxx WORKA = YES -BLKSIZE บอกความยาวของ I/O area เป็นจำนวนไบท์ -DEVADDR คือ SYS ที่กำหนดใน Extent -DEVICE คือการกำหนดรุ่นของแผ่นจานแม่เหล็กที่ใช้

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
		- Organization บอกการจัดองค์การของแฟ้มข้อมูล - name ทำหน้าที่บอกชื่อของแฟ้มข้อมูล ข. FILE SECTION โดยกำหนดที่ FD ซึ่งมีรูปแบบดังนี้ FD file-name BLOCK CONTAINS clause RECORD CONTAINS clause RECORDING MODE clause LABEL RECORD clause - filename เป็นชื่อของแฟ้มข้อมูล ซึ่งจะต้องตรงกับที่กำหนดไว้ใน SELECT - Clause ทำหน้าที่บอกถึงโครงสร้างและลักษณะของแฟ้มข้อมูล ซึ่งบางวลีอาจจะเขียนหรือไม่เขียนก็ได้ แต่ LABEL RECORD clause จำเป็นต้องเขียนเสมอ	- IOAREA 1 คือชื่อพื้นที่ I/O ที่ใช้ - RECFORM บอกลักษณะของแฟ้มข้อมูลที่ใช้ - RECSIZE บอกขนาดของระเบียนข้อมูล -TYPEFLE บอกลักษณะของแฟ้มข้อมูล

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	๓. ผู้ใช้ไม่ต้องกำหนดรายละเอียดต่าง ๆ ของระเบียนข้อมูล เมื่อต้องการอ่านหรือบันทึกข้อมูลจึงค่อยกำหนดโดยอ้างถึงในรูปของ FORMAT	๓. ผู้ใช้ต้องกำหนดรายละเอียดต่าง ๆ ของระเบียนข้อมูล เช่น ชื่อของระเบียนข้อมูลหรือความยาวและประเภทของข้อมูลในแต่ละข้อมูลย่อย ซึ่งสามารถอธิบายด้วยวลีต่าง ๆ ที่เป็นอิสระต่อกัน โดยมีรูปแบบที่ใช้ดังนี้ - level number { data-name / FILLER } [PICTURE Clause] [VALUE Clause] [USAGE Clause] [REDEFINES Clause] [OCCURS Clause] [SYNCHRONIZED Clause] [JUSTIFIED Clause] [BLANK WHEN ZERO Clause] [SIGN Clause]	๓. ผู้ใช้ต้องกำหนดรายละเอียดต่าง ๆ ของระเบียนข้อมูล เช่นเดียวกับโคบอล โดยกำหนดใน DS ซึ่งมีรูปแบบดังนี้ NAME DS dtLn - NAME คือ ชื่อของระเบียนข้อมูลหรือชื่อของข้อมูลย่อย - d คือ จำนวนข้อมูลย่อยที่ใช้ - t คือ รูปแบบของข้อมูลที่จะเก็บ - L คือ ความยาว - n คือ จำนวนไบท์

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	๔. ในการกำหนดความยาวและลักษณะของข้อมูลในแต่ละข้อมูลย่อยจะกำหนดในรูปของ FORMAT ซึ่งแบ่งได้ดังนี้ ก. <u>ตัวเลข</u> แบ่งเป็นดังนี้ ๑. <u>เลขจำนวนเต็ม</u> ใช้กำหนดด้วย I FORMAT ๒. <u>ทศนิยม</u> ใช้กำหนดด้วย D,E และ F FORMAT ๓. <u>เลขฐาน ๑๖</u> ใช้กำหนดด้วย Z FORMAT	- ใช้สำหรับอธิบายรายละเอียดของระเบียนข้อมูลใน - FILE SECTION - WORKING-STORAGE SECTION - LINKAGE SECTION ๔. ในการกำหนดความยาวจะกำหนดโดยใช้ PICTURE ซึ่งจะใช้สัญญลักษณ์ 9 สำหรับข้อมูลที่เป็นตัวเลข สำหรับข้อมูลที่ใช้เป็นตัวอักษรสามารถกำหนดโดยใช้ A หรือ X ก็ได้ ซึ่ง X นี้ สามารถใช้ในการกำหนดข้อมูลที่เป็นตัวเลขได้ด้วย แต่ตัวเลขที่กำหนดโดยใช้ X นี้ไม่สามารถนำไปใช้ในการคำนวณได้สำหรับการกำหนดลักษณะที่ข้อมูลต่าง ๆจะถูกเอาไปเก็บไว้ในหน่วยความจำหลักของเครื่องคอมพิวเตอร์กำหนดโดยใช้ USAGE ซึ่งมีรูปแบบดังนี้	๔.ในการกำหนดความยาวของข้อมูลในแต่ละข้อมูลย่อย จะกำหนดในรูปของ t ซึ่งสามารถแบ่งประเภทได้ดังนี้ - B ใช้กำหนดตัวเลขที่เป็นเลขฐาน ๒ - X ใช้กำหนดตัวเลขที่เป็นเลขฐาน ๑๖ - P ใช้กำหนดค่า Packed decimal - C ใช้กำหนดตัวอักษร - F กำหนดค่าเป็น FULL WORD - H กำหนดค่าเป็น HALF WORD - D กำหนดค่าเป็น DOUBLE WORD

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
		[USAGE IS] { DISPLAY / COMPUTATIONAL / COMPUTATIONAL-1 / COMPUTATIONAL-2 / COMPUTATIONAL-3 } - DISPLAY ใช้กำหนดตัวอักษร (Character) ซึ่งจะใช้เนื้อที่ ๑ ไบท์ ต่อหนึ่งตัวอักษร - COMPUTATIONAL ใช้กำหนดตัวเลขที่เป็นเลขฐาน ๒ ซึ่งจะเป็น FULLWORD หรือ HALFWORD ขึ้นอยู่กับจำนวนหลักของเลข 9 หลัง PICTURE - COMPUTATIONAL-1 ใช้สำหรับตัวเลขที่เป็น floating-point ซึ่งกินเนื้อที่ ๔ byte (Single precision) - COMPUTATIONAL-2 ใช้สำหรับตัวเลขแต่จะใช้เนื้อที่ ๘ ไบท์ (Double precision) ซึ่งจะทำให้ผลลัพธ์ที่ได้จากการคำนวณถูกต้องมากยิ่งขึ้น	

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	นอกจาก FORMAT ต่าง ๆดังที่กล่าวมาแล้วนี้ ยังมี FORMAT ที่ผู้ใช้สามารถอ้างถึงได้อีกดังนี้ ๑. G FORMAT ใช้กับข้อมูลทั่วไปที่เป็นเลขจำนวนเต็ม ทศนิยม หรือ ข้อมูลทางตรรก โดยขึ้นกับชนิดของตัวแปรใน I/O list ๒. L FORMAT ใช้ในการโยกย้ายข้อมูลทางตรรก ๓. H FORMAT ใช้ในการกำหนดค่าตัวอักษร (Literal) ๔. X FORMAT ใช้ในการข้ามขอบเขตข้อมูลในระเบียนข้อมูลทาง INPUT และใช้ในการเว้นว่างทาง OUTPUT	- COMPUTATIONAL -3 ใช้สำหรับ Packed decimal	

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	๔. ไม่สามารถใช้สัญญลักษณ์เพื่อกำหนดรูปแบบรายการย่อยของข้อมูล (ตัวเลข) ที่จะพิมพ์ออกมาเพื่อให้อยู่ในรูปแบบที่อ่านง่าย สวยงาม และมีความหมายในการบัญชี (Edit form)	๔. สามารถใช้สัญญลักษณ์เพื่อกำหนดรูปแบบรายการย่อยของข้อมูล (ตัวเลข) ที่จะพิมพ์ออกมาเพื่อให้อยู่ในรูปแบบที่อ่านง่าย สวยงาม และมีความหมายในการบัญชี สัญญลักษณ์ต่าง ๆ ที่ใช้มี ๑๑ ตัว คือ * Z Ø B , . ₿ + - DB และ CR โดยแบ่งออกเป็น ๒ พวก ดังนี้ ก. <u>fix Insertion</u> เป็นการเพิ่มสัญญลักษณ์ที่กำหนดลงไปในจำนวนเลขที่จะพิมพ์ออกมา เช่น CR DB และ ₿ เป็นต้น ข. <u>Replacement</u> เป็นการแทนที่กลุ่มของเลขที่อยู่ข้างหน้าของจำนวนเลขที่จะพิมพ์ออกมาด้วยสัญญลักษณ์ที่กำหนด เช่น * , ₿ หรือใช้แทนด้วยช่องว่าง	๔. สามารถกำหนดรูปแบบรายการย่อยของข้อมูล (ตัวเลข) ที่จะพิมพ์ออกมาเพื่อให้อยู่ในรูปแบบที่อ่านง่าย สวยงาม และมีความหมายในการบัญชี เช่นเดียวกับแบบโคบอล

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๒. คำสั่ง READ	มีรูปแบบดังนี้ READ(a,b, ERR=C, END = d) list - a เป็น data set reference number ซึ่งจะต้องมีอยู่เสมอ - b เป็นหมายเลขประจำคำสั่งของคำสั่ง FORMAT ที่ใช้ควบคุมคำสั่ง READ นี้หรือเป็นเนื้อที่ (Array) ที่ใช้เก็บ FORMAT หรือเป็น NAMELIST ส่วน b อาจจะมีหรือไม่มีก็ได้ - C เป็นหมายเลขประจำคำสั่งประเภท executable ในโปรแกรมเดียวกัน การทำงานของโปรแกรมจะย้ายจากคำสั่ง READ ไปยังคำสั่งที่มีหมายเลขประจำคำสั่ง C ในกรณีที่การโยกย้ายข้อมูลมีข้อผิดพลาด C นี้อาจจะมีหรือไม่มีก็ได้	มีรูปแบบดังนี้ READ file-name Record [INTO identifier] AT END imperative Statement - file-name เป็นชื่อของแฟ้มข้อมูลซึ่งจะต้องตรงกับที่กำหนดไว้ใน SELECT - INTO identifier อาจจะมีหรือไม่มีก็ได้ เมื่อใช้คำว่า INTO ระเบียนข้อมูลนั้นจะถูกอ่านไปเก็บในเนื้อที่ที่บอกด้วย IDENTIFIER ซึ่งกำหนดรายละเอียดต่าง ๆ ไว้ใน WORKING-STORAGE SECTION - AT END เมื่อคอมพิวเตอร์ทำงานจนกระทั่งระเบียนข้อมูลหมดแล้ว ก็จะไปทำงานยังคำสั่งที่ตามหลังคำว่า AT END	มีรูปแบบดังนี้ GET {filename/(1)} [,workname/,(0)] - file-name เป็นชื่อของแฟ้มข้อมูลซึ่งต้องตรงกับที่กำหนดไว้ใน DS - Workname คือ Output Work area ที่กำหนดด้วย คำสั่ง DS

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	- d เป็นหมายเลขประจำคำสั่งประเภท executable ในโปรแกรมเดียวกัน การทำงานของโปรแกรมจะย้ายจากคำสั่ง READ ไปยังคำสั่งที่มีหมายเลขประจำคำสั่ง d ในกรณีที่ข้อมูลที่อ่านหมด - list เป็น I/O list ซึ่งอาจมีหรือไม่มีก็ได้		
ก. คำสั่ง WRITE	มีรูปแบบดังนี้ WRITE (a,b) list - a เป็น data set reference number ซึ่งต้องมีอยู่เสมอ - b เป็นหมายเลขประจำคำสั่งของคำสั่ง FORMAT ที่ใช้แสดงลักษณะของระเบียนข้อมูลที่จะบันทึก หรือเป็นเนื้อที่ที่ใช้เก็บ FORMAT หรือเป็น NAMELIST - list เป็น I/O list ซึ่งอาจมีหรือไม่มีก็ได้	มีรูปแบบดังนี้ WRITE record-name [from identifier-1] INVALID KEY imperative-Statement - from identifier-1 จะมีหรือไม่มีก็ได้ ถ้ามีหมายความว่าให้บันทึกระเบียนข้อมูลที่เก็บในเนื้อที่ที่บอกด้วย identifier ซึ่งกำหนดรายละเอียดต่าง ๆ ไว้ใน WORKING-STORAGE SECTION	มีรูปแบบดังนี้ PUT {filename/(1)} [,workname/,(0)] - file-name เป็นชื่อของแฟ้มข้อมูลซึ่งต้องตรงกับที่กำหนดไว้ใน DS - Workname คือ Output work area ที่กำหนดด้วยคำสั่ง DS

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
. คำสั่ง CLOSE	- ไม่มีคำสั่ง CLOSE	CLOSE [file-name UNIT [WITH LOCK]] - ประโยคคำสั่งนี้ทำหน้าที่หยุดการทำงานของเครื่องอ่าน/บันทึกข้อมูล โดยขณะบอกให้หยุดทำงาน อาจมีคำสั่งให้ทำงานพิเศษควบคู่ไปด้วย เช่น LOCK - UNIT ใช้สำหรับแผ่นจานแม่เหล็ก - LOCK เมื่อใช้คำว่า LOCK นี้ หัวอ่าน/บันทึกข้อมูลจะยกออกจากแผ่นจานแม่เหล็ก	CLOSE [filename] - ประโยคคำสั่งนี้ทำหน้าที่หยุดการทำงานของเครื่องอ่าน/บันทึกข้อมูล เช่นเดียวกับแบบโคบอล - filename คือชื่อของแฟ้มข้อมูลที่ใช้
. คำสั่ง OPEN	-ไม่มีคำสั่ง OPEN	OPEN { [INPUT {file-name}] / [OUTPUT {file-name}] / [I-O {file-name}] } -file-name เป็นชื่อของแฟ้มข้อมูลที่เขียนตามหลังคำว่า FD ใน FILE-SECTION	OPEN [file-name] - file-name คือชื่อของแฟ้มข้อมูลที่ใช้

๔.๒.๑.๓ COMPILE AND EXECUTION TIME

จากตารางที่ ๔.๒.๑.๓ จะเห็นได้ว่าภาษาแอสเซมเบอร์ใช้เวลาในการแปลคำสั่งภาษาที่เขียนขึ้นให้เป็นคำสั่งภาษาเครื่อง(Compilation time) มากที่สุด รองลงมาได้แก่ภาษาโคบอล และภาษาฟอร์แทรนใช้เวลาน้อยที่สุด การที่ภาษาแอสเซมเบอร์ใช้เวลาในการแปลคำสั่งภาษาที่เขียนขึ้นให้เป็นคำสั่งภาษาเครื่องมากที่สุด เนื่องมาจาก Macro level ของตัวภาษาแอสเซมเบอร์เอง นอกจากนี้ภาษาแอสเซมเบอร์ยังสามารถกำหนดชนิดแบบต่าง ๆ ได้มาก ทำให้เสียเวลาในการแปลคำสั่ง (ภาคผนวก ช.)

สำหรับเวลาที่คอมพิวเตอร์ใช้ทำงานตามที่นักเขียนโปรแกรมสั่งไว้ (Execution time) ภาษาแอสเซมเบอร์จะใช้เวลาน้อยที่สุด เนื่องมาจากว่าในการเขียนโปรแกรมแอสเซมเบอร์นั้น สามารถใช้คำสั่ง (Instruction) ที่มีประสิทธิภาพและเหมาะสมได้ตามที่ต้องการ ภาษาโคบอลจะใช้เวลามากกว่าแอสเซมเบอร์ แต่น้อยกว่าฟอร์แทรน เพราะว่าภาษาโคบอลไม่คล่องตัวเท่ากับภาษาแอสเซมเบอร์ แต่ก็สามารถจัดระเบียนข้อมูลให้เป็นบล็อคได้ และในการเขียนโปรแกรมสามารถบังคับได้ถึงระดับที่เป็นไบท์ ส่วนภาษาฟอร์แทรนใช้เวลามากที่สุด เนื่องจากว่าฟอร์แทรนไม่สามารถจัดระเบียนข้อมูลให้เป็นบล็อคได้จึงต้องเสียเวลาสำหรับอินท์พุท/เอาท์พุท (I/O time) มาก และภาษาเครื่องที่แปลจากฟอร์แทรนมีประสิทธิภาพต่ำกว่า

ตารางที่ ๔.๒.๑.๓ COMPILE AND EXECUTION TIME ของการจัดแฟ้มข้อมูลแบบแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
COMPILE TIME (สร้าง)	.๐๓ วินาที	.๐๖ วินาที	.๒๖ วินาที
EXECUTION TIME	๒.๔๖ วินาที	.๐๖ วินาที	.๐๔ วินาที
COMPILE TIME (ดึงข้อมูล)	.๑๐ วินาที	.๒๔ วินาที	.๓๑ วินาที
EXECUTION TIME	.๓๔ วินาที	.๐๓ วินาที	.๐๑ วินาที

๕.๒.๑.๔ FILE PROTECTION AND ERROR MESSAGE

จากตารางที่ ๕.๒.๑.๔ จะเห็นว่าภาษาโคบอลมีคำสั่ง Checkpoint/Restart ทำให้สามารถเก็บสถานะภาพของข้อมูลต่าง ๆ ในหน่วยความจำหลัก ณ จุดใดจุดหนึ่งตามที่นักเขียนโปรแกรมกำหนด ในขณะที่ทำงานได้ ดังนั้นกรณีที่เกิดการขัดข้องในการประมวลผลก็ไม่จำเป็นต้องย้อนกลับมาทำงานใหม่ตั้งแต่ต้น คือสามารถย้อนไปยังจุด (check point) ที่ใกล้กับการขัดข้องของการประมวลผลมากที่สุดและดำเนินการประมวลผลต่อไป (Restart) ได้ แต่ภาษาฟอร์แทรนไม่มีคำสั่งเพื่อให้สามารถที่จะทำเช่นนี้ได้

ตารางที่ ๕.๒.๑.๔ FILE PROTECTION AND ERROR MESSAGE ของการจัดแฟ้มข้อมูลแบบแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	- สามารถใช้ label เพื่อกันความผิดพลาดได้โดยกำหนด logical unit ที่ DLBL Statement ใน JCL (ภาคผนวก ง)	- สามารถใช้ label เพื่อกันความผิดพลาดได้โดยกำหนดชื่อที่ DLBL Statement ใน JCL (ภาคผนวก ง) - สามารถเก็บสถานภาพของข้อมูลต่าง ๆ ในหน่วยความจำหลักในขณะที่กำลังทำงานได้โดยใช้คำสั่ง RERUN ON System-name EVERY integer RECORDS OF file-name	สามารถใช้ Label เพื่อกันความผิดพลาดได้โดยกำหนดชื่อที่ DLBL Statement JCL เช่นเดียวกับโคบอล

๕.๒.๒ การเปรียบเทียบภาษาฟอร์แทรน โคบอล และ แอสเซมเบอร์ กับการจัดแฟ้มข้อมูลแบบ แดม (DAM)

๕.๒.๒.๑ ลักษณะการจัดแฟ้มข้อมูล และขอบเขตความสามารถ

ในการพิจารณานี้จะแยกตามหัวข้อต่าง ๆ ดังนี้

ก. ลักษณะแฟ้มข้อมูล

ข. ลักษณะระเบียนข้อมูล

ค. เนื้อที่ที่ใช้ในการเก็บระเบียนข้อมูล

จากตารางที่ ๕.๒.๒.๑ จะเห็นได้ว่าภาษาโคบอลและแอสเซมเบอร์มีความคล่องตัวในการจัดแฟ้มข้อมูลมากกว่าภาษาฟอร์แทรน ดังจะเห็นว่า ระเบียนข้อมูลที่ใช้กับภาษาโคบอลและแอสเซมเบอร์นั้น สามารถจัดให้อยู่ในลักษณะใดก็ได้ กล่าวคือระเบียนข้อมูล ข้อมูลอาจมีความยาวคงที่ ไม่คงที่ หรือเป็นแบบ Span ก็ได้ และระเบียนข้อมูลอาจถูกจัดให้เป็นบล็อคหรือไม่เป็นบล็อคก็ได้ ในขณะที่ภาษาฟอร์แทรนนั้น ความยาวของระเบียนข้อมูลต้องคงที่ และไม่สามารถจัดให้เป็นบล็อคได้ จึงทำให้มีแพคกิงเดนซิตี้ต่ำ ซึ่งจะมีผลต่อประสิทธิภาพในการทำงาน (ภาคผนวก ข)

ตารางที่ ๔.๒.๒.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถในการจัดแฟ้มข้อมูลแบบแคม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑. ลักษณะแฟ้มข้อมูล	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๒.๑ คือเป็นแบบ Relative กับตำแหน่งที่เริ่มต้น	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๒.๑	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๒.๑
๒. ลักษณะระเบียนข้อมูล	ก. ต้องมีความยาวคงที่ ข. ไม่สามารถจัดให้เป็นบล็อคได้	ก. อาจมีความยาวคงที่ ไม่คงที่หรือแบบ Span ได้ ข. อาจจัดให้เป็นบล็อคหรือไม่เป็นบล็อคก็ได้ แต่โดยปกตินิยมใช้กับระเบียนข้อมูลที่ไม่เป็นบล็อค	มีลักษณะเช่นเดียวกับแบบโคบอล
๓. เนื้อที่ที่ใช้ในการเก็บระเบียนข้อมูล	ในกรณีที่ระเบียนข้อมูลไม่เรียงต่อเนื่องกัน (คือเป็นลักษณะที่กระโดด) จะมีเนื้อที่ว่างที่ไม่ได้ใช้มาก	ในกรณีที่จัดระเบียนข้อมูลเป็นบล็อคจะใช้เนื้อที่ในการเก็บระเบียนข้อมูลน้อยกว่าแบบไม่จัดเป็นบล็อค	เช่นเดียวกับโคบอล

๕.๒.๒.๒ ลักษณะการใช้โปรแกรม

ในการพิจารณาจะแยกพิจารณาตามหัวข้อดังนี้ คือ

ก. การเขียนโปรแกรม

ข. คำสั่ง READ

ค. คำสั่ง WRITE

ง. คำสั่ง CLOSE

จ. คำสั่ง OPEN.

จากตารางที่ ๕.๒.๒.๒ จะเห็นได้ว่าการใช้โปรแกรมของภาษาฟอร์แทรน สามารถใช้ง่ายและสะดวกกว่าภาษาโคบอลและแอสเซมเบอร์ ดังจะเห็นว่าในการกำหนดลักษณะแฟ้มข้อมูลนั้นก็สามารถกำหนดได้โดยใช้คำสั่ง DEFINE FILE ซึ่งเมื่อกำหนดแล้วผู้ใช้สามารถอ้างถึงแล้วนำไปใช้ได้ทันที ไม่จำเป็นต้องกำหนดหลาย ๆ คำสั่ง เช่นเดียวกับโคบอล ส่วนภาษาแอสเซมเบอร์นั้นมีความยุ่งยากในการกำหนด DTF มากกว่าโคบอลและฟอร์แทรน แต่ฟอร์แทรนก็มีข้อเสียดังที่กล่าวแล้วในหัวข้อ ๕.๒.๑.๒

ตารางที่ ๕.๒.๒.๒ ลักษณะการใช้โปรแกรมของการจัดแฟ้มข้อมูลแบบแดม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑.การเขียน โปรแกรม	๑. เขียนโปรแกรมสั้น (ภาคผนวก ฉ) ๒. ผู้ใช้ต้องกำหนดลักษณะของแฟ้มข้อมูล โดยใช้คำสั่ง DEFINE FILE ซึ่งมีรูปแบบดังนี้ DEFINE FILE a(m,r,f,v) - a เป็นเลขเต็มบวกที่ใช้เป็น Data set reference number ซึ่งตรงกับ DOS logical Unit name Assign ไปยังแผ่นจานแม่เหล็กและชื่อ DOS ที่จะกำหนดลงใน DLBL - m เป็นจำนวนมากที่สุดของระเบียนข้อมูล ข้อมูลที่จะใส่เข้าไปหรือดึงออกมาใช้ใน แฟ้มข้อมูลที่ขึ้นอยู่กับ a - r บอกขนาดที่มากที่สุดในแต่ละระเบียน ระเบียนข้อมูล ๓ ลักษณะคือ	๑. เขียนโปรแกรมยาว (ภาคผนวก ฉ) ๒. ผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลใน INPUT-OUTPUT SECTION เช่นเดียวกับแบบแซม แต่มีสิ่งที่ต้องกำหนดเพิ่มคือ ก. ACCESS IS {RANDOM / SEQUENTIAL} ACTUAL KEY IS data-name -data-name คือชื่อของข้อมูลย่อยที่ใช้เป็นคีย์ในระเบียนข้อมูลของแฟ้มข้อมูลแบบเข้าถึงข้อมูลโดยตรง	๑. ผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อ ที่ DTF macro filename DTFDA BLKSIZE =nnnn DEVICE = nnnn ERRBYTE =xxxxxxxx IOAREA1 =xxxxxxxx TYPEFLE =xxxxxxxx AFTER =YES DEVADDR =SYSnnn DSKXTNT =n KEYARG =xxxxxx KEYLEN =nnn RECFORM =xxxxxxxx RELTYPE =xxx SEEKADR =xxxxxxxx WRITEKY =YES

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
	ก. จำนวน Storage location เป็นไบท์ ข. จำนวนตัวอักษรเป็นไบท์ ค. จำนวน Storage unitเป็น word - f เป็นตัวกำหนดว่า การอ่าน/บันทึก จะมี FORMAT หรือไม่ โดยกำหนดอักษรต่อไปนี้ L:การอ่าน/บันทึก จะมีหรือไม่มี FORMAT ก็ได้ ขนาดที่มากที่สุดของแต่ละระเบียนข้อมูล นับเป็น Storage location (ไบท์) E:การอ่าน/บันทึก ต้องมี FORMAT ขนาดมากที่สุดของแต่ละระเบียนข้อมูล นับเป็นตัวอักษร (ไบท์) U:การอ่าน/บันทึก ไม่มี FORMAT ขนาดมากที่สุดของแต่ละระเบียนข้อมูล นับเป็น Storage Unit (Word) - V ทำหน้าที่คล้ายดัชนีชี้ไปยังระเบียนข้อมูลถัดไปจากระเบียนข้อมูลที่ถูกอ่าน/บันทึก ผ่านไปแล้ว		-BLKSIZE บอกความยาวของ I/O area DEVICE กำหนดรุ่นของแผ่นจานแม่เหล็กที่ใช้ -ERRBYTE เป็นการตรวจสอบเงื่อนไขต่าง ๆ ที่เกิด -IOAREA1 ใช้กำหนดชื่อ I/O areaของแฟ้มข้อมูล -TYPEFLE บอกลักษณะของแฟ้มข้อมูลว่าเป็น INPUT หรือ OUTPUT -DEVADDR คือ Symbolic Unit -DSKXTNT คือจำนวนของ Extent ที่ใช้ -KEYARG คือชื่อของคีย์ -KEYLEN คือความยาวของคีย์ที่ใช้ -RECFORM บอกลักษณะของแฟ้มข้อมูลว่าเป็นแบบใด -RELTYPE กำหนด relative address ว่าเป็นแบบ HEXA หรือ DECIMAL -SEEKADR คือชื่อของแทรคที่อ้างถึง

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๒. คำสั่ง READ	READ(a'r,b,ERR=c) list - a เป็น data set reference number ตัวเดียวกันกับค่าใน DEFINE FILE - r เป็นตำแหน่งทางกายภาพของระเบียนข้อมูลที่จะอ่าน หมายเลขของ FORMAT (กรณีอ่านแบบมี FORMAT) - c เป็นหมายเลขของคำสั่งที่จะให้ไปเมื่อเกิดข้อผิดพลาดขึ้น	READ file name RECORD INTO identifier INVALID KEY imperative-statement - ลักษณะการใช้เช่นเดียวกับที่กล่าวในไอแซม	name READ {filename, KEY/ID} - filename เป็นชื่อของแฟ้มข้อมูลที่กำหนดที่ DTF macro
๓. คำสั่ง WRITE	WRITE (a'r,b) list - a: data set reference number เป็นตัวเดียวกันกับค่าใน DEFINE FILE - r: เป็นตำแหน่งทางกายภาพของระเบียนข้อมูลที่จะบันทึก - b: หมายเลขของ FORMAT (กรณีบันทึกแบบมี FORMAT)	WRITE record-name [FROM identifier-1] INVALID KEY imperative-Statement - มีลักษณะการใช้เช่นเดียวกับที่กล่าวแล้วในไอแซม	name WRITE {filename/(1)},{KEY/ID/ AFTER} -filename เป็นชื่อของแฟ้มข้อมูลที่กำหนดที่ DTF macro -KEY, ID,AFTER เป็น macro ที่ส่งระเบียนข้อมูลจาก VS ไปยัง DASD Storage

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๔. คำสั่ง CLOSE	- ไม่มีคำสั่ง นอกจากคำสั่งต่าง ๆ ดังกล่าวมาแล้ว ผู้ใช้ยังสามารถใช้คำสั่ง FIND เพื่อลดเวลาหอบสนอง (response time) ในการทำงาน เมื่อใช้คำสั่งนี้ เครื่องจะหาระเบียนข้อมูลต่อไปไว้เลย เมื่อถึงเวลาจะอ่านเข้ามาจะได้อ่านทันที ไม่ต้องเสียเวลาหาระเบียนข้อมูลอีก คำสั่ง FIND มีรูปแบบดังนี้ FIND (a'r) - a เป็น data set reference number ซึ่งเป็นตัวเดียวกับที่กำหนดใน DEFINE FILE - r เป็น relative position ที่ต้องการหา	- มีลักษณะเช่นเดียวกับที่กล่าวในแซม	- มีลักษณะเช่นเดียวกับที่กล่าวในแซม
๕. คำสั่ง OPEN	- ไม่มีคำสั่ง OPEN	OPEN $\left\{\begin{array}{l}\left[\text{INPUT}\ \{\text{file-name}\}\right]\\ \left[\text{OUTPUT}\ \{\text{file-name}\}\right]\\ \left[\text{I-O}\ \{\text{file-name}\}\right]\end{array}\right\}$ -file-name เป็นชื่อของแฟ้มข้อมูลที่เขียนตามหลังคำว่า FD ใน FILE-SECTION	OPEN [file-name] - file-name คือชื่อของแฟ้มข้อมูลที่ใช้

๕.๒.๒.๓ COMPILE AND EXECUTION TIME

จากตารางที่ ๕.๒.๒.๓ จะเห็นได้ว่าแอสเซมเบอร์ใช้เวลาในการเปลี่ยนคำสั่งภาษาที่เขียนขึ้นให้เป็นคำสั่งภาษาเครื่อง (Compilation time) มากที่สุด รองลงมาได้แก่ ภาษา โคบอล และภาษาฟอร์แทรนใช้เวลาน้อยที่สุด การที่ภาษาแอสเซมเบอร์ ใช้เวลาในการแปลคำสั่งภาษาที่เขียนขึ้นให้เป็นคำสั่งภาษาเครื่องมากที่สุด เนื่องมาจาก Macro level ของตัวภาษาแอสเซมเบอร์เอง นอกจากนี้ภาษาแอสเซมเบอร์ยังสามารถกำหนดรูปแบบต่าง ๆ ได้มาก ทำให้เสียเวลาในการแปลคำสั่ง

สำหรับ เวลาที่คอมพิวเตอร์ใช้ทำงานตามที่นักเขียนโปรแกรมสั่งไว้ (execution time) ภาษาแอสเซมเบอร์จะใช้เวลาน้อยที่สุด เนื่องมาจากว่าในการเขียนโปรแกรมแอสเซมเบอร์นั้นสามารถใช้คำสั่งที่มีประสิทธิภาพและเหมาะสมได้ตามที่ต้องการ ส่วนภาษาฟอร์แทรนนั้นใช้เวลามากที่สุด เนื่องจากว่าฟรอ์แทรนไม่สามารถจัดระเบียนข้อมูลให้เป็นบล็อคได้ จึงต้องเสียเวลาสำหรับไอโอ (I/O time) มาก และภาษาเครื่องที่แปลจากฟอร์แทรนมีประสิทธิภาพต่ำกว่า

ตารางที่ ๕.๒.๒.๓ COMPILE AND EXECUTION TIME ของการจัดแฟ้มข้อมูลแบบแคม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
COMPILE TIME (สร้าง)	.๐๔ วินาที	.๐๘ วินาที	.๓๑ วินาที
EXECUTION TIME	.๓๕ วินาที	.๓๒ วินาที	.๐๑ วินาที
COMPILE TIME	.๐๒ วินาที	.๐๕ วินาที	.๒๙ วินาที
EXECUTION TIME	.๐๖ วินาที	.๐๑ วินาที	.๐๑ วินาที
COMPILE TIME	.๐๒ วินาที	.๐๖ วินาที	.๒๙ วินาที
EXECUTION TIME	๑.๔๖ วินาที	.๑๒ วินาที	.๐๕ วินาที
COMPILE TIME (ดึงข้อมูล)	.๑๐ วินาที	.๒๘ วินาที	.๓๙ วินาที
EXECUTION TIME	.๑๐ วินาที	.๐๒ วินาที	.๐๑ วินาที

๕.๒.๒.๔ FILE PROTECTION AND ERROR MESSAGE

จากตารางที่ ๕.๒.๒.๔ จะเห็นได้ว่าภาษาโคบอลสามารถ handle I/O error ที่เกิดขึ้นได้โดยการใช้ INVALID KEY หรือโดยการกำหนด DECLARATIVE

ตารางที่ ๕.๒.๒.๕ FILE PROGECTION AND ERROR MESSAGE ของการจัดแฟ้มข้อมูลแบบแคม

<table>
<tr><th></th><th>ฟอร์แทรน</th><th>โคบอล</th><th>แอสเซมเบอร์</th></tr>
<tr>
<td></td>
<td>- สามารถใช้ lebel เพื่อกันความผิดพลาดได้โดยกำหนด logical unit ที่ DLBL Statement ใน JCL</td>
<td>- สามารถใช้ label เพื่อกันความผิดพลาดได้โดยกำหนดชื่อที่ DLBL Statement ใน JCL
- ผู้ใช้สามารถ handle I/O error ที่เกิดขึ้นได้จาก
ก. การใช้ INVALID KEY ซึ่งข้อผิดพลาดต่าง ๆ ที่เกิดขึ้นจะเป็นไปตามตารางดังนี้
<table>
<tr><th>ACESS</th><th>OPEN</th><th>I-O VERB</th><th>เงื่อนไข</th></tr>
<tr><td>SEQ</td><td>OUTPUT</td><td>WRITE</td><td>ตำแหน่งของแทรคเกินค่าที่กำหนดใน EXTENT</td></tr>
<tr><td>RANDOM</td><td>INPUT</td><td>READ</td><td>ไม่พบระเบียนข้อมูลที่ต้องการ</td></tr>
<tr><td></td><td>OUTPUT</td><td>WRITE</td><td>ตำแหน่งของแทรคเกินค่าที่กำหนดใน EXTENT</td></tr>
<tr><td></td><td>I-O</td><td>READ
REWRITE</td><td>ตำแหน่งของแทรคเกินค่าที่กำหนดใน EXTENT</td></tr>
</table>
</td>
<td>-สามารถใช้ label เพื่อกันความผิดพลาดได้โดยกำหนดชื่อที่ DLBL Statement ใน JCL เช่นเดียวกับโคบอล</td>
</tr>
</table>

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์

ข. ใช้ DECLARATIVE ข้อผิดพลาดต่าง ๆ จะเป็นไปตาม ตารางดังนี้

ACCESS	OPEN	I-O VERB	เงื่อนไข
SEQ	INPUT	READ	WRONG LENGTH RECORD 2 DATA CHECK IN COUNT AREA 1 DATA CHECK FOR KEY AND/OR DATA 4
RANDOM	INPUT I-O	READ	เหมือนแบบ SEQ
	OUTPUT	WRITE	WRONG LENGTH RECORD 2 DATA CHECK IN COUNT AREA 1 DATA CHECK FOR KEY AND/OR DATA 4 NO ROOM FOUND 3
	I-O	REWRITE	WRONG LENGTH RECORD 2 DATA CHECK IN COUNT AREA 1 DATA CHECK IN KEY AND/OR DATA 4

๔.๒.๓ การเปรียบเทียบภาษาโคบอล และแอสเซมเบอร์กับการจัดแฟ้มข้อมูลแบบไอแซม (ISAM) สำหรับในวิธีนี้ไม่สามารถใช้กับภาษาฟอร์แทรนได้

๔.๒.๓.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถ

ในการพิจารณานี้จะแยกตามหัวข้อต่าง ๆ ดังนี้

ก. ลักษณะแฟ้มข้อมูล

ข. ลักษณะระเบียนข้อมูล

ค. เนื้อที่ที่ใช้ในการเก็บระเบียนข้อมูล

จากตารางที่ ๔.๒.๓.๑ จะเห็นได้ว่าระเบียนข้อมูลที่ใช้กับภาษาโคบอลและแอสเซมเบอร์นั้น อาจถูกจัดให้เป็นบล็อคหรือไม่เป็นบล็อคก็ได้ แต่ระเบียนข้อมูลที่ใช้นั้นจะต้องมีความยาวคงที่

ตารางที่ ๔.๒.๓.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถในการจัดแฟ้มข้อมูลแบบไอแซม

	ฟอร์แทรน	โคบอล	เอสแซมเบอร์
๑. ลักษณะแฟ้มข้อมูล ๒. ลักษณะระเบียนข้อมูล ๓. เนื้อที่ที่ใช้ในการเก็บระเบียนข้อมูล ระเบียนข้อมูล	-	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๓.๑ ก. ระเบียนข้อมูลต้องมีความยาวคงที่ ข. อาจถูกจัดให้เป็นบล็อคหรือไม่เป็นบล็อคก็ได้ ในกรณีจัดระเบียนข้อมูลเป็นบล็อค จะใช้เนื้อที่ในการเก็บระเบียนข้อมูลน้อยกว่าแบบไม่จัดเป็นบล็อค เนื่องจากว่าสามารถลดจำนวนเนื้อที่ระหว่างระเบียนข้อมูลได้	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๓ ก. มีลักษณะเช่นเดียวกับโคบอล ข. เช่นเดียวกับโคบอล เช่นเดียวกับโคบอล

๕.๒.๓.๒ ลักษณะของการใช้โปรแกรม

ในการพิจารณานี้จะแยกตามหัวข้อต่าง ๆ ดังนี้

ก. การเขียนโปรแกรม

ข. คำสั่ง READ

ค. คำสั่ง WRITE

ง. คำสั่ง REWRITE

จ. คำสั่ง CLOSE

ฉ. คำสั่ง OPEN

จากตารางที่ ๕.๒.๓.๒ จะเห็นได้ว่า การใช้โปรแกรมของภาษาโคบอลและแอสเซมเบอร์ นั้นผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูล รวมทั้งรายละเอียดต่าง ๆ ของระเบียนข้อมูลไว้ก่อน ซึ่งการกำหนดนี้ภาษาแอสเซมเบอร์มีการกำหนดยุ่งยากมากกว่า ดังจะเห็นได้จากการกำหนด DTF

ตารางที่ ๕.๒.๓.๒ ลักษณะการใช้โปรแกรมของการจัดแฟ้มข้อมูลแบบไอแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑. การเขียนโปรแกรม		๑. ผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลใน INPUT-OUTPUT SECTION เช่นเดียวกับแบบแซม แต่อย่างไรก็ตามมีสิ่งที่ต้องกำหนดเพิ่มคือ ก. ACCESS IS { SEQUENTIAL / RANDOM } - กรณีที่เป็นการสร้าง Access จะต้องเป็น SEQUENTIAL เสมอ - กรณีที่เป็นการดึงข้อมูลมาใช้ หรือการแก้ไขเปลี่ยนแปลง อาจเป็นแบบ SEQUENTIAL หรือ RANDOM ก็ได้ ข. RECORD KEY IS data-name - data-name คือชื่อของข้อมูลย่อยที่ใช้เป็นคีย์ในระเบียนข้อมูลของแฟ้มข้อมูลแบบดัชนี ค. NOMINAL KEY IS data-name NOMINAL KEY นี้จะกำหนดก็ต่อเมื่อ ผู้ใช้ต้องการอ่านหรือบันทึกข้อมูลแฟ้มข้อมูลแบบดัชนี สำหรับใน FILE SECTION นั้นจะต้องกำหนดดังนี้คือ	๑. ผู้ใช้ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลใน DTF MACRO file-name DTFIS DSKXTNT =n IOROUT =xxxxxx KEYLEN =nnn NRECDS =nnn HINDEX =nnnn RECFORM =xxxxxxxx RECSIZE =nnnn DEVICE =nnnn IOAREAL =xxxxxxxx KEYLOC =nnnn WORKL =xxxxxxxx - DSKXTNT คือจำนวนสูงสุดของ EXTENT ที่ใช้ - IOROUT เป็นตัวกำหนดลักษณะของแฟ้มข้อมูลว่าเป็นการสร้าง ดึงข้อมูลมาใช้หรือเปลี่ยนแปลงแ ข้อมูล

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
		- RECORDING MODE IS F - LABEL RECORD IS STANDARD ๒. ผู้ใช้ต้องกำหนดรายละเอียดต่าง ๆ ของระเบียนข้อมูล เช่นเดียวกับแซม และมีหลักเกณฑ์ในการกำหนดเช่นเดียวกับแซม	- KEYLEN คือความยาวของคีย์คิดเป็นไบท์ - NRECDS คือจำนวนของระเบียนข้อมูลใน ๑ บล็อค - HINDEX คือการกำหนดว่าจะเก็บดัชนีสูงสุดไว้ที่ใด - RECFORM คือ การกำหนดลักษณะของระเบียนข้อมูลว่าเป็นแบบบล็อคหรือไม่เป็นบล็อค - RECSIZE คือจำนวนตัวอักษรในระเบียนข้อมูลทางตร - DEVICE คือ การกำหนดว่าจะเก็บพื้นที่หลักและพื้นที่ส่วนเกินไว้ที่ใด - IOAREAL คือการกำหนดพื้นที่ที่ใช้ว่าเป็นการสร้างหรือการเพิ่มเติมแฟ้มข้อมูล - KEYLOC คือตำแหน่งของคีย์ที่ใช้ - WORKL คือการกำหนดพื้นที่สำหรับระเบียนข้อมูลซึ่งชื่อที่กำหนดนี้จะต้องตรงกับที่กำหนดใน DS ๒. ผู้ใช้ต้องกำหนดรายละเอียดต่าง ๆ ของระเบียนข้อมูลใน DS เช่นเดียวกับแบบ แซม ๓. กำหนดความยาวและประเภทของข้อมูลในแต่ละข้อมูลย่อยในรูปของ t เช่นเดียวกับแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๒. คำสั่ง READ		READ file-name RECORD [INTO identifier] INVALID KEY imperative-Statement - file-name identifier มีวิธีใช้เช่นเดียวกับที่ได้อธิบายแล้วในแซม - INVALID KEY ทำหน้าที่ตรวจสอบดูว่ามีระเบียนข้อมูลใด ๆ ในแฟ้มข้อมูลนั้นที่มีคีย์เหมือนกับคีย์ที่เก็บไว้ใน NOMINAL KEY หรือไม่ ถ้าไม่มีก็จะทำให้ให้เกิดเงื่อนไขที่เรียกว่า INVALID KEY แล้วคอมพิวเตอร์จะข้ามไปทำงานตามประโยคคำสั่งที่เขียนไว้หลังคำว่า INVALID KEY	READ {file-name/(1)} , KEY - file-name คือชื่อของแฟ้มข้อมูลซึ่งต้องตรงกับที่กำหนดไว้ที่ DTF macro - KEy คือค่าคีย์ที่ใช้ในการค้นหา ซึ่งผู้ใช้ต้องกำหนดใน KEYARG
๓. คำสั่ง WRITE		WRITE recordname [FROM identifier-1] INVALID KEY imperative-Statement - record-name เป็นชื่อของระเบียนข้อมูลทางตรรกที่เขียนไว้ใน File Section - FROM หมายถึงให้ย้ายข้อมูลจากเนื้อที่ในหน่วยความจำหลักที่ชื่อว่า identifier ไปไว้ที่ ๆ ชื่อว่า record-name แล้วทำการพิมพ์หรือบันทึกระเบียนข้อมูลบนแฟ้มข้อมูลที่ระบุ	WRITE {file-name/(1)} , NEW KEY - file-name คือชื่อของแฟ้มข้อมูลซึ่งต้องตรงกับที่กำหนดไว้ที่ DTF macro - NEWKEy คือ ค่าคีย์ที่ใช้ในการบันทึก

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
		- INVALID KEY จะทำหน้าที่ตรวจสอบดูว่าค่าคีย์ของระเบียนข้อมูล ที่บันทึกลงไปมีค่าซ้ำกันหรือไม่ ถ้ามีค่าซ้ำกันแล้วคอมพิวเตอร์จะข้ามไปทำงานตามประโยคคำสั่งที่เขียนไว้หลังคำว่า INVALID KEY	
๔. คำสั่ง REWRITE		REWRITE record-name [FROM identifier] INVALID KEY imperative-Statement - ใช้เมื่อต้องการเปลี่ยนแปลงแก้ไขข้อมูล - ลักษณะการใช้คำสั่งนี้เหมือนกับคำสั่ง WRITE	WRITE {file-name /(1)}, KEY - file-name คือชื่อของแฟ้มข้อมูล ซึ่งต้องตรงกับที่กำหนดใน DS - KEY คือตำแหน่งที่จะทำการเปลี่ยนแปลงแก้ไข โดยจะกำหนดใน KEYARG
๕. คำสั่ง CLOSE		มีลักษณะเช่นเดียวกับที่กล่าวแล้วในแซม	มีลักษณะเช่นเดียวกับในแซม
๖. คำสั่ง OPEN	- ไม่มีคำสั่ง OPEN	OPEN { [INPUT {file-name}] [OUTPUT {file-name}] [I-O {file-name}] } - file-name เป็นชื่อของแฟ้มข้อมูลที่เขียนตามหลังคำว่า FD ใน FILE-SECTION	OPEN [file-name] - file-name คือชื่อของแฟ้มข้อมูลที่ใช้

๕.๒.๓.๓ COMPILE AND EXECUTION TIME

จากตารางที่ ๕.๒.๓.๓ ภาษาแอสเซมเบอร์ใช้เวลาในการแปลคำสั่งภาษาที่เขียนขึ้นให้เป็นคำสั่งภาษาเครื่อง มากกว่าภาษาโคบอล เนื่องมาจาก Macro level ของตัวภาษาแอสเซมเบอร์เอง นอกจากนี้ภาษาแอสเซมเบอร์ยังสามารถกำหนดรูปแบบต่าง ๆ ได้มาก ทำให้เสียเวลาในการแปลคำสั่ง

สำหรับเวลาที่คอมพิวเตอร์ใช้ทำงานตามที่นักเขียนโปรแกรมสั่งไว้ ภาษาแอสเซมเบอร์นั้นจะใช้เวลาน้อยกว่า เนื่องมาจากว่าในการเขียนโปรแกรมแอสเซมเบอร์สามารถใช้คำสั่งที่มีประสิทธิภาพและเหมาะสมได้ตามที่ต้องการ

ตารางที่ ๕.๒.๓.๓ COMPILE AND EXECUTION TIME ของการจัดแฟ้มข้อมูลแบบไอแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
COMPILE TIME (สร้าง)		.๔๐ วินาที	.๓๐ วินาที
EXECUTION TIME		.๓๐ วินาที	.๐๒ วินาที
COMPILE TIME		.๐๕ วินาที	.๒๙ วินาที
EXECUTION TIME		.๐๑ วินาที	.๐๑ วินาที
COMPILE TIME		.๐๖ วินาที	.๒๙ วินาที
EXECUTION TIME		.๑๒ วินาที	.๐๕ วินาที
COMPILE TIME (ดึงข้อมูล)		.๒๘ วินาที	.๓๒ วินาที
EXECUTION TIME		.๐๒ วินาที	.๐๑ วินาที

๕.๒.๓.๔ FILE PROTECTION AND ERROR MESSAGE

จากตารางที่ ๕.๒.๓.๔ จะเห็นได้ว่า ภาษาโคบอลนั้นสามารถ handle I/O error ที่เกิดขึ้นได้ง่ายกว่าภาษาแอสเซมเบอร์ โดยสามารถใช้ INVALID KEY หรือโดยกำหนด DECLARATIVE ส่วนภาษาแอสเซมเบอร์นั้นจะใช้ในรูปของ TM(Test mask) ซึ่งผู้ใช้จะต้องเขียนคำสั่งขึ้นเพื่อนำมาใช้ตรวจสอบความผิดพลาดขึ้นเอง

ตารางที่ ๕.๒.๓.๔ FILE PROTECTION AND ERROR MESSAGE ของการจัดแฟ้มข้อมูลแบบไอแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
		- สามารถใช้ label เพื่อกันความผิดพลาดได้ โดยกำหนดชื่อที่ DLBL Statement ใน JCL (ภาคผนวก จ) - ผู้ใช้สามารถ handle I/O errorที่เกิดขึ้นได้จาก ก. การใช้ INVALID KEY ซึ่งข้อผิดพลาดต่างๆ ที่เกิดขึ้นเป็นไปได้ดังตารางนี้	- สามารถใช้ label เพื่อกันความผิดพลาดได้ โดยกำหนดชื่อที่ DLBL Statement ใน JCL ผู้ใช้สามารถทราบข้อผิดพลาดที่เกิดขึ้นได้จากการใช้ TM ซึ่งข้อผิดพลาดต่าง ๆ จะเป็นไปดังนี้

ACCESS	OPEN	I-O VERB	CONDITION	บิท	ค่า Hexa	ข้อผิดพลาด
SEQ	INPUT	START	ไม่พบระเบียนข้อมูล	๐	X'80'	DASD ERROR
			ที่ต้องการ	๑	X'40'	WRONG LENGTH RECORD
	OUTPUT	WRITE	ระเบียนข้อมูลซ้ำกัน	๒	X'20'	PRIME DATA AREA FULL
RANDOM	INPUT	READ	ไม่พบระเบียนข้อมูล	๓	X'10'	CYLINDER INDEX FULL
	I-O	REWRITE	ที่ต้องการ	๔	X'08'	MASTER INDEX FULL
	I-O	WRITE	ระเบียนข้อมูลซ้ำกัน	๕	X'04'	DUPLICATE RECORD
				๖	X'02'	SEQUENCE CHECK
				๗	X'01'	PRIME DATA AREA OVERFLA

	ฟอร์แทรน	โคบอล				แอสเซมเบอร์
		ข. ใช้ DECLARATIVE ซึ่งบอกข้อผิดพลาดดังนี้				
		ACCESS	OPEN	I-O VERB	CONDITION	ไบท์
		SEQ	INPUT	READ	DASD ERROR	1
			I-O	REWRITE	WRONG LENGTH RECORD	2
				START	DASD ERROR	1
			OUTPUT	WRITE	DASD ERROR	1
					WRONG LENGTH RECORD	2
					PRIME DATA FULL	3
					CYLINDER INDEX FULL	4
					MASTER INDEX FULL	5
		RANDOM	INPUT	READ	DASD ERROR	1
			I-O	REWRITE	WRONG LENGTH RECORD	2
			I-O	WRITE	DASD ERROR	1
					WRONG LENGTH RECORD	2
					OVERFLOW AREA FULL	6

๕.๒.๔ เปรียบเทียบภาษาโคบอล และแอสเซมเบอร์ กับการจัดแฟ้มข้อมูลแบบวีแซม (VSAM)

๕.๒.๔.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถ

จากตารางที่ ๕.๒.๔.๑ จะเห็นได้ว่าระเบียนข้อมูลที่ใช้กับภาษาโคบอลและแอสเซมเบอร์นั้นสามารถจัดให้อยู่ในลักษณะใดก็ได้ กล่าวคือระเบียนข้อมูลอาจมีความยาวคงที่ไม่คงที่ หรือเป็นแบบ Span ก็ได้ และระเบียนข้อมูลอาจถูกจัดเป็นบล็อคหรือไม่เป็นบล็อคก็ได้

ตารางที่ ๕.๒.๔.๑ ลักษณะการจัดแฟ้มข้อมูลและขอบเขตความสามารถในการจัดแฟ้มข้อมูลแบบวีแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑. ลักษณะแฟ้มข้อมูล ๒. ลักษณะระเบียนข้อมูล		มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๔.๑ ก. มีความยาวคงที่ ไม่คงที่ หรือเป็นแบบ Span ก็ได้ ข. อาจถูกจัดให้เป็นบล็อค หรือไม่เป็นบล็อคก็ได้	มีลักษณะเช่นเดียวกับที่กล่าวในหัวข้อ ๓.๒.๔.๑ ก. เช่นเดียวกับแบบโคบอล ข. เช่นเดียวกับแบบโคบอล

๕.๒.๔.๒ ลักษณะการใช้โปรแกรม

จากตารางที่ ๕.๒.๔.๒ จะเห็นได้ว่าการใช้โปรแกรมของภาษาโคบอลและแอสเซมเบอร์นั้นผู้ใช้ต้องกำหนดโครงสร้าง และลักษณะของแฟ้มข้อมูล รวมทั้งรายละเอียดต่าง ๆ ของระเบียนข้อมูลไว้ก่อน ซึ่งการกำหนดนี้ภาษาแอสเซมเบอร์มีการกำหนดที่ยุ่งยากมากกว่า ดังจะเห็นได้จากการกำหนด ACB macro และ RPL macro

ตารางที่ ๔.๒.๔.๒ ลักษณะการใช้โปรแกรมของการจัดแฟ้มข้อมูลแบบวีแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๑. การเขียนโปรแกรม		๑. ผู้ใช้ต้องกำหนด MASTER CATALOG, SPACE และ CLUSTER ก่อน (ภาคผนวก ค) ๒. ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลใน INPUT-OUTPUT SECTION เช่นเดียวกับแบบแซม แต่มีสิ่งที่ต้องกำหนดเพิ่มคือ ก. ORGANIZATION IS INDEXED ข. ACCESS IS { SEQUENTIAL / RANDOM / DYNAMIC } ค. RECORD KEY IS data-name สำหรับใน FILE-SECTION นั้นก็กำหนดเช่นเดียวกับในแซม	๑. ผู้ใช้ต้องกำหนด MASTER CATALOG, SPACE และ CLUSTER เช่นเดียวกับแบบโคบอล ๒. ต้องกำหนดโครงสร้างและลักษณะของแฟ้มข้อมูลใ ก. ACB macro name ACB DDNAME = flename MACRF = (option / option) PASSWD = addr -filename คือชื่อของแฟ้มข้อมูลซึ่งต้องตรงกับใน DLBL -option คือการกำหนดว่าจะดึงข้อมูลโดยวิธีใด เช่นถ้ากำหนดเป็น KEY แสดงว่าต้องการดึงข้อมูลโดยใช้คีย์ -addr คือการกำหนด PASSWORD ที่ใช้โดยตรงกับชื่อที่กำหนดไว้ที่ DC ข.RPL macro

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
		๓. ผู้ใช้ต้องกำหนดลักษณะของระเบียนข้อมูลที่ใช้เช่นเดียวกับที่กล่าวแล้วในแซม	name RPL ACB = addr AM = VSAM AREA = Addr AREALEN = n OPTCD = (Option / option) - ACB = Addr คือการกำหนดตำแหน่งของ ACB - AREA = Addr คือการกำหนดตำแหน่งของ I/O work area ที่ใช้สำหรับคำสั่ง GET, PUT - n คือการกำหนดความยาวของ work area - option คือการกำหนดวิธีที่จะดึงข้อมูล ๓. ผู้ใช้ต้องกำหนดลักษณะของระเบียนข้อมูลที่ใช้เช่นเดียวกับที่กล่าวแล้วในแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
๒. คำสั่ง READ		READ file-name RECORD [INTO identifier] INVALID KEY imperative Statement มีวิธีใช้เช่นเดียวกับที่กล่าวใน ไอแซม ในกรณีที่ใช้การ Access แบบ Dynamic คำสั่ง READ ที่ใช้จะเป็น READ filename RECORD [INTO identifier] NEXT RECORD AT END imperative Statement	-name GET RPL = {addr/(1)} -addr คือตำแหน่งของ RPL
๓. คำสั่ง WRITE		WRITE record-name from identifier-1 INVALID KEY imperative Statement - มีวิธีการใช้เช่นเดียวกับที่กล่าวในไอแซม	name PUT RPL = {addr/(1)} - addr คือตำแหน่งของ RPL
๔. คำสั่ง CLOSE		มีรูปแบบและวิธีใช้เช่นเดียวกับที่กล่าวในไอแซม	มีรูปแบบและวิธีใช้เช่นเดียวกับที่กล่าวในไอแซม
๕. คำสั่ง OPEN	- ไม่มีคำสั่ง OPEN	OPEN {[INPUT {file-name}] [OUTPUT {file-name}] [I-O {file-name}]} - file-name เป็นชื่อของแฟ้มข้อมูลที่เขียนตามหลังคำว่า FD ใน FILE-SECTION	OPEN [file-name] - file-name คือชื่อของแฟ้มข้อมูลที่ใช้

๕.๒.๔.๓ COMPILE AND EXECUTION TIME

จากตารางที่ ๕.๒.๔.๓ ภาษาแอสเซมเบอร์ใช้เวลาในการแปลคำสั่งภาษาที่เขียนขึ้นให้เป็นคำสั่งภาษาเครื่องมากกว่าภาษาโคบอล เนื่องมาจาก Macro level ของตัวภาษาแอสเซมเบอร์เอง นอกจากนี้ภาษาแอสเซมเบอร์ยังสามารถกำหนดรูปแบบต่าง ๆ ได้มากทำให้เสียเวลาในการแปลคำสั่ง

สำหรับเวลาที่คอมพิวเตอร์ใช้ทำงานตามที่นักเขียนโปรแกรมสั่งไว้ ภาษาแอสเซมเบอร์จะใช้เวลาน้อยกว่า เนื่องมาจากว่าในการเขียนโปรแกรมแอสเซมเบอร์นั้น สามารถใช้คำสั่งที่มีประสิทธิภาพและเหมาะสมได้ตามที่ต้องการ

ตารางที่ ๕.๒.๔.๓ COMPILE AND EXECUTION TIME ของการจัดแฟ้มข้อมูลแบบวีแซม

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์
COMPILE TIME (สร้าง)		.๐๗ วินาที	.๓๔ วินาที
EXECUTION TIME		.๑๕ วินาที	.๑๓ วินาที
COMPILE TIME		.๐๕ วินาที	.๓๘ วินาที
EXECUTION TIME		.๐๖ วินาที	.๐๕ วินาที
COMPILE TIME		.๐๕ วินาที	.๓๘ วินาที
EXECUTION TIME		.๒๔ วินาที	.๒๓ วินาที
COMPILE TIME (ดึงข้อมูล)		.๒๘ วินาที	.๓๖ วินาที
EXECUTION TIME		.๐๖ วินาที	.๐๓ วินาที

๕.๒.๔.๔ FILE PROTECTION AND ERROR MESSAGE

จากตารางที่ ๕.๒.๔.๔ จะเห็นได้ว่าทั้งภาษาโคบอลและแอสเซมเบอร์ สามารถตรวจสอบข้อผิดพลาดที่เกิดขึ้นโดยการตรวจสอบสถานภาพของแฟ้มข้อมูลได้เช่นเดียวกัน

ตารางที่ ๕.๒.๔.๔ FILE PROTECTION AND ERROR MESSAGE ของการจัดแฟ้มข้อมูลแบบวีแซม

<table>
<tr><th></th><th>ฟอร์แทรน</th><th>โคบอล</th><th>แอสเซมเบอร์</th></tr>
<tr><td></td><td></td><td>-สามารถใช้ label เพื่อกันความผิดพลาดได้โดยกำหนดชื่อที่ DLBL ใน JCL
-สามารถทราบข้อผิดพลาดที่เกิดขึ้นได้โดยการตรวจสอบสถานะภาพของแฟ้มข้อมูล
-(File Status) ซึ่งดูได้จากตารางดังนี้
ก. สถานะภาพของแฟ้มข้อมูล เมื่อ OPEN
<table>
<tr><th>สถานะภาพ</th><th>สาเหตุที่เกิด</th></tr>
<tr><td>๓๐</td><td>ผิดพลาดทาง I-O</td></tr>
<tr><td>๙๑</td><td>PASSWORD ที่ใช้ไม่ถูกต้อง</td></tr>
<tr><td>๙๒</td><td>ผิดพลาดทางตรรก</td></tr>
<tr><td>๙๓</td><td>ไม่สามารถหาแฟ้มข้อมูลที่ต้องการได้ ซึ่งอาจเกิดจากว่าแฟ้มข้อมูลนั้น ๆ ถูก OPEN ไปแล้ว</td></tr>
<tr><td>๙๕</td><td>ไม่พบแฟ้มข้อมูลนี้ในแคตตาลอค</td></tr>
<tr><td>๙๖</td><td>ไม่มี DLBL</td></tr>
</table></td><td>- สามารถใช้ label เพื่อกันความผิดพลาดได้โดยกำหนดชื่อ ที่ DLBL ใน JCL
- สามารถทราบข้อผิดพลาดที่เกิดขึ้นได้โดยการตรวจสอบสถานะภาพของแฟ้มข้อมูลโดยการใช้ REGISTER ตัวที่ ๑๕ ซึ่งทราบข้อผิดพลาดที่เกิดขึ้นได้โดยดูจากตารางเช่นกัน (๕)</td></tr>
</table>

	ฟอร์แทรน	โคบอล	แอสเซมเบอร์

ข. สถานะภาพของแฟ้มข้อมูลที่ REGUEST TIME

สถานะภาพ	สาเหตุที่เกิด
๐๐	ไม่มีข้อผิดพลาด
๑๐	คำสั่ง READ ที่อ่านแบบเรียงลำดับ ไปพบ EOF
๒๑	เกิด Sequence error
๒๒	ระเบียนข้อมูลที่ต้องการเพิ่มมีอยู่แล้วในแฟ้มข้อมูล
๒๓	ระเบียนข้อมูลที่ต้องการอ่าน หรือเปลี่ยนแปลง แก้ไข หรือ ลบทิ้ง ไม่มีในแฟ้มข้อมูล
๒๔	เตรียมพื้นที่ที่จะใช้บันทึกข้อมูลไว้ไม่พอ (สำหรับ KSDS)
๓๐	เกิดข้อผิดพลาดทาง I-O
๓๔	เตรียมพื้นที่ที่จะใช้บันทึกข้อมูลไว้ไม่พอ (สำหรับ (ESDS)
๙๒	เกิดข้อผิดพลาดทางตรรก
๙๔	ไม่มีตัวชี้บอกจุดเริ่มต้นที่จะอ่าน เมื่อใช้คำสั่ง READ แบบเรียงลำดับ