บทที่ 3

การออกแบบตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล

การพัฒนาตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอลนั้น เป็นขั้นตอนที่จำลองส่วนของตัวแปลโปรแกรมภาษาโคบอล มาใช้ประโยชน์ เพื่อตรวจสอบความถูกต้องของการเขียนโปรแกรมในภาษาดังกล่าว หรืออาจกล่าวอีกนัยหนึ่งว่า ตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล หรือโคบอล พรี-คอมไพเลอร์ (COBOL Pre-Compiler) คือ ซอฟต์แวร์ที่สร้างขึ้นโดยอาศัยหลักการสร้างตัวแปลโปรแกรม ในส่วนของ การวิเคราะห์เลกซิเคิล และ ส่วนของพาสเซอร์ นั่นเอง

การออกแบบเพื่อสร้างซอฟต์แวร์ดังกล่าวนั้น ได้แบ่งขั้นตอนการพัฒนาออกเป็นเฟสได้ 3 เฟส (3 Phases) ได้แก่

1. <u>เฟสการวิเคราะห์เลกซิเคิล</u> (Lexical Analysis Phase) ทำหน้าที่สร้างหน่วยย่อยของสายวลีต่าง ๆ เพื่อนำไปใช้ในเฟสต่อไป เราเรียกหน่วยย่อยของสายวลีดังกล่าวว่า โทเกน (Token)* นอกจากนั้นแล้วในเฟสนี้ หากพบความผิดพลาดต่าง ๆ จะสร้างรหัสความผิด (Error Code) เพื่อเก็บไว้ใช้ในเฟสต่อไปด้วย
2. <u>เฟสการทำพาสซิง</u> (Parsing Phase) มีหน้าที่ตรวจสอบความถูกต้องของโปรแกรมภาษาโคบอล โดยรับข้อมูลโทเกนที่ได้จากเฟสการวิเคราะห์เลกซิเคิลมาทำการวิเคราะห์ความถูกต้องดังกล่าว ซึ่งถ้าพบความผิดพลาด จะสร้างรหัสความผิดพลาดไว้เช่นเดียวกัน
3. <u>เฟสการสร้างข่าวสารความผิดพลาด</u> (Error Message Generation) เป็นเฟสที่ทำหน้าที่ถอดรหัสความผิดพลาดที่ได้จาก เฟสการวิเคราะห์เลกซิเคิล และเฟสการทำพาสซิง ให้เป็นข่าวสารที่ผู้เขียนโปรแกรมสามารถอ่าน และทำความ

---

* โทเกน (Token) เป็นสายวลีที่เล็กที่สุด ที่ยังคงความหมายในตัวเอง ในความหมายของการออกแบบตัวตรวจสอบภาษาโคบอล เพื่อตรวจสอบการเรียงลำดับของคำแต่ละคำ

เข้าใจในที่ผิดของโปรแกรมได้

ในแต่ละเฟสดังกล่าวข้างต้นนั้น เพื่อความง่ายและชัดเจนต่อความเข้าใจ จะนำเสนอการออกแบบการสร้างแต่ละเฟส ภายใต้หัวข้อต่าง ๆ ดังต่อไปนี้ คือ

1. นิยามของปัญหา (Problem Definition)
2. ข้อมูลที่จำเป็นในการสร้าง (Data Structure)
3. รูปแบบและโครงสร้างของข้อมูลที่ใช้ (Format of Data)
4. วิธีการที่ใช้ในการแก้ปัญหา (Algorithm)
5. แนวทางการแบ่งส่วนของระบบเป็นส่วนย่อย (Modularization)

<u>เฟสที่ 1 เฟสการวิเคราะห์เลกซิเคิล</u>

<u>นิยามของปัญหา</u>

ในเฟสที่ 1 นี้ เป็นส่วนเริ่มต้นของการออกแบบทั้งหมด โดยแบ่งหน้าที่ต่าง ๆ ออกเป็น

1. ทำหน้าที่อ่านข้อมูลตั้งต้น อันหมายถึงโปรแกรมดิบภาษาโคบอลของผู้ใช้งาน เพื่อนำมาสู่การประมวลผล ในเรื่องนี้ เราจะพบปัญหาที่เกิดขึ้น กล่าวคือ ข้อมูลโปรแกรมดิบภาษาโคบอลของผู้ใช้งาน ถูกเตรียมขึ้นโดยเครื่องไมโครคอมพิวเตอร์ "ไทยทำ" ซึ่งใช้ซอฟต์แวร์ในการเตรียมข้อมูล* (Data Entry) ทำการสร้างโปรแกรมลงในแผ่นจานแม่เหล็ก (Diskette) ขนาด 8 นิ้ว โดยซอฟต์แวร์ดังกล่าวทำหน้าที่จำลองเครื่องไมโครคอมพิวเตอร์เป็นเสมือนกับเครื่องเตรียมข้อมูลลงบนแผ่นจานแม่เหล็กไอบีเอ็ม (IBM Key-to-Diskette) ดังนั้นแผ่นจานแม่เหล็กที่ผู้ใช้งาน เตรียมโปรแกรมดิบภาษาโคบอลนั้น จะถูกจัดเตรียมขึ้นเป็นรูปแบบที่เรียกว่า IBM-3470 Format**

---

* เป็นโปรแกรมเพื่อใช้ในการเตรียมข้อมูลลงในแผ่นจานแม่เหล็กขนาด 8 นิ้ว ที่ทางสถาบันบริการคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย จัดสร้างขึ้น

** เป็นลักษณะวิธีการจัดเนื้อที่ในแผ่นจานแม่เหล็กขนาด 8 นิ้ว ของบริษัท ไอบีเอ็ม

ในส่วนของตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอลที่สร้างขึ้นนั้น สร้างภายใต้ระบบปฏิบัติการสำหรับไมโครคอมพิวเตอร์ ซีพีเอ็ม/80 (CP/M-80) ซึ่งมีส่วนที่สำคัญเกี่ยวกับการจัดการแผ่นจานแม่เหล็กที่เรียกว่าบีดอส (BDOS) อันทำหน้าที่จัดการแผ่นจานแม่เหล็กที่มีรูปแบบการบันทึก ข้อมูลในลักษณะ CP/M Format

ปัญหาเบื้องต้นที่พบคือ เราไม่สามารถเรียกส่วน บีดอส นั้นให้ทำหน้าที่อ่านข้อมูลโปรแกรมดิบที่ผู้ใช้เตรียมขึ้นด้วยซอฟต์แวร์เตรียมข้อมูล ซึ่งเก็บบันทึกในรูปแบบ IBM-3470 Format ได้ เนื่องจากการจัดโครงสร้างข้อมูลในแผ่นจานแม่เหล็กต่างกัน ดังนั้นจึงต้องมีการนำส่วนประกอบบางส่วนของซอฟต์แวร์เตรียมข้อมูล มาดัดแปลงเพื่อใช้อ่านโปรแกรมดิบภาษาโคบอลเข้าสู่การประมวลผลของระบบต่อไป

2. คัดลอกเนื้อหาของโปรแกรมดิบที่อ่านได้นั้น ลงมาเก็บในแฟ้มข้อมูลมาตรฐานของระบบ โดยแฟ้มข้อมูลนั้นประกอบด้วย

2.1 เลขลำดับบรรทัด (Line Number) ที่ใช้กำกับแต่ละบรรทัดของโปรแกรมดิบเพื่อใช้อ้างอิงในการให้ข่าวสารความผิดพลาดของโปรแกรม

2.2 คำสั่งในภาษาโคบอลที่ผู้ใช้เขียนเป็นโปรแกรม

3. ตรวจสอบตัวอักษรที่ปรากฏในคอลัมน์ 7 ของแต่ละบรรทัดว่าเป็นตัวอักษรที่ถูกต้องหรือไม่ ทั้งนี้ตัวอักษรที่ถูกต้องที่สามารถปรากฏได้ในคอลัมน์ที่ 7 คือ

3.1 ตัวอักษร * หมายถึง บรรทัดหมายเหตุ (Comment Line) จะไม่นำเอาสายอักขระใด ๆ ที่ปรากฏในบรรทัดนั้น เข้าสู่การประมวลผล

3.2 ตัวอักษร - หมายถึง บรรทัดนั้นเป็นการต่อบัตรจากบรรทัดก่อนหน้านั้น อันจะมีผลในการต่อค่าคงที่ตัวอักษร (Non-Numeric Literal)

3.3 ตัวอักษร / หมายถึง ให้ตัวแปลโปรแกรมภาษาโคบอล สร้างรหัสควบคุม*

---

* สำหรับเครื่องพิมพ์ทั่วไป รหัส 0BH หมายถึงการขึ้นหน้ากระดาษใหม่ (Form Feed) ส่วนเครื่องพิมพ์ประเภททีละบรรทัด (Line Printer) จะใช้ ตัวอักษร 1 ในคอลัมน์ที่ 1

(Control Character Code) ลงในแฟ้มข้อมูลแสดงผลเพื่อพิมพ์ (Print file - SYSPRINT) เพื่อให้เครื่องพิมพ์ (Line Printer) ขึ้นหน้ากระดาษใหม่

3.4 ตัวอักษร D หมายถึง คำสั่งในบรรทัดนั้น เป็นคำสั่งช่วยในการหาที่ผิดแบบทีละบรรทัด (Debugging Line)

3.5 ตัวอักษรว่าง (Blank) หมายถึงบรรทัดเป็นคำสั่งปรกติของภาษาโคบอล

4. สร้างโทเกนจากโปรแกรมดิบภาษาโคบอลที่อ่านเข้ามา โดยตรวจสอบความผิดพลาดบางอย่าง ที่อาจตรวจพบได้ในเฟสนี้ โดยมักจะเป็นเรื่องที่เกี่ยวกับเครื่องหมายต่าง ๆ และเครื่องหมายวรรคตอน

4.1 เครื่องหมาย . (Period) ไม่ควรมีช่องว่างนำหน้า

4.2 เครื่องหมาย (จะต้องมีช่องว่างนำหน้า แต่ต้องไม่มีช่องว่างตามหลัง

4.3 เครื่องหมาย) จะต้องไม่มีช่องว่างนำหน้า แต่ต้องมีช่องว่างตามหลัง

4.4 เครื่องหมาย , และเครื่องหมาย ; ต้องไม่มีช่องว่างนำหน้า แต่ต้องมีช่องว่างตามหลัง

4.5 เครื่องหมาย + และเครื่องหมาย - ต้องมีช่องว่างนำหน้า แต่อาจจะมีช่องว่างตามหลังหรือไม่มีก็ได้ ในกรณีที่ไม่มีช่องว่างตามหลัง ตัวที่ตามมาต้องเป็นตัวเลข กล่าวคือ เป็นลักษณะค่าคงที่ตัวเลขที่มีเครื่องหมาย (Unary Sign)

4.6 ในกรณีค่าคงที่ตัวเลขต้องตรวจความถูกต้องเป็นไปได้ ดังนี้

4.6.1 มีเครื่องหมาย + หรือเครื่องหมาย - ได้เพียงตัวเดียว

4.6.2 มีเครื่องหมายพิเศษ (Special Character) อื่น ๆ ได้เพียงเครื่องหมาย . แสดงจุดทศนิยมเพียงอย่างเดียว และเพียงตัวเดียวใน 1 ค่าคงที่ตัวเลขเท่านั้น

4.7 ตรวจสอบเลขแสดงลำดับคำสั่ง (Serial Number) ในคอลัมน์ที่ 1-6 ว่าเรียงลำดับถูกต้องหรือไม่

5. ในการวิเคราะห์เลกซิเคิลนั้น การสร้างโทเกน จะไม่เก็บเครื่องหมายวรรคตอนต่าง ๆ เข้าสู่แฟ้มข้อมูลโทเกน เพื่อใช้ในเฟสต่อไป

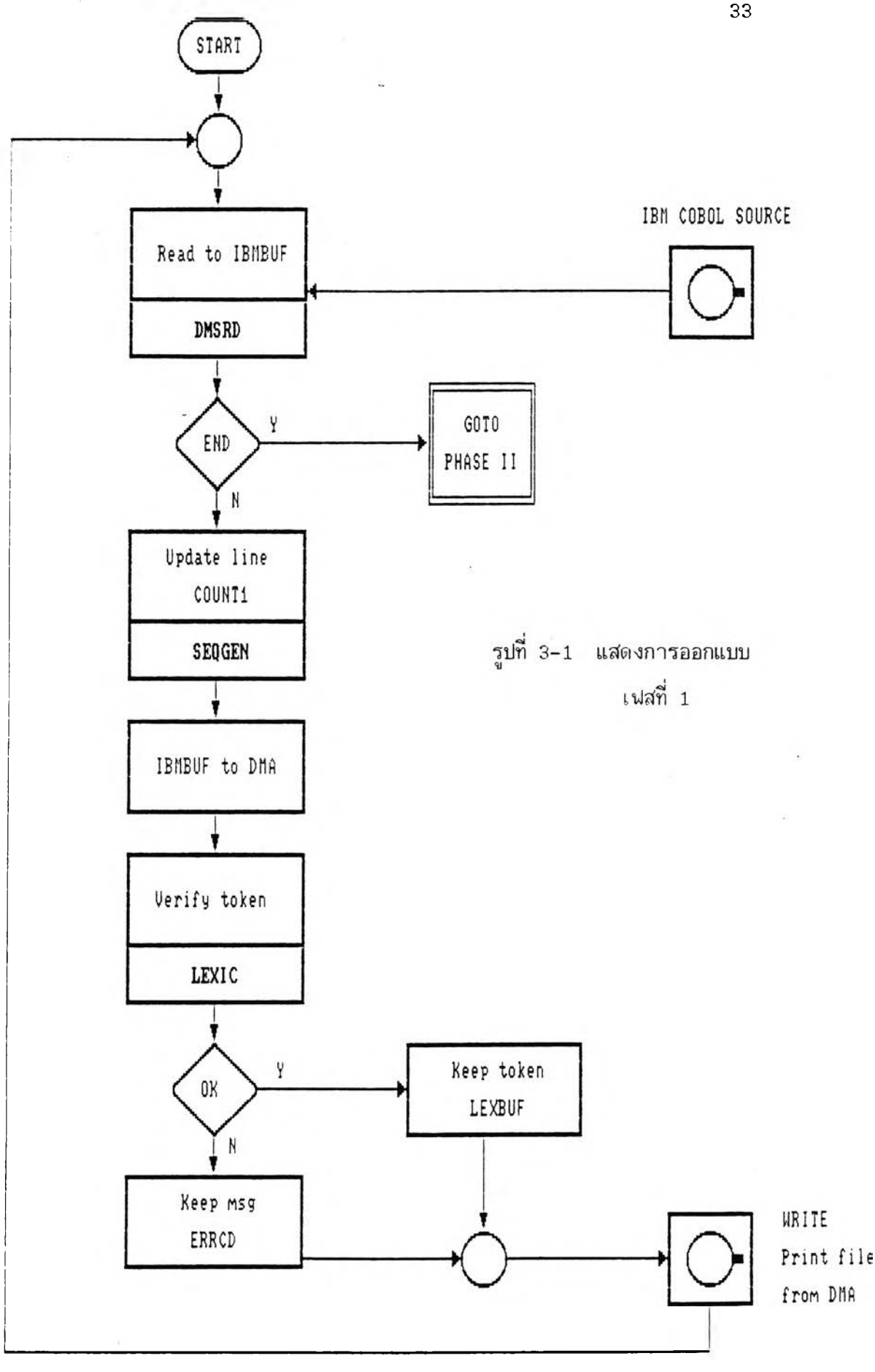
ด้วยกฎเกณฑ์ต่าง ๆ ดังกล่าวในเฟสนี้ ถ้าพบความผิดพลาดจะสร้างรหัสแสดงความผิดพลาดเก็บบันทึกลงในแฟ้มข้อมูล เพื่อนำไปใช้ในเฟสที่ 3

หน้าที่ประการสำคัญ คือการสร้างโทเกน เก็บบันทึกไว้ในแฟ้มข้อมูล เพื่อนำไปใช้ในการวิเคราะห์ความถูกต้องของไวยากรณ์ภาษา ในเฟสที่ 2

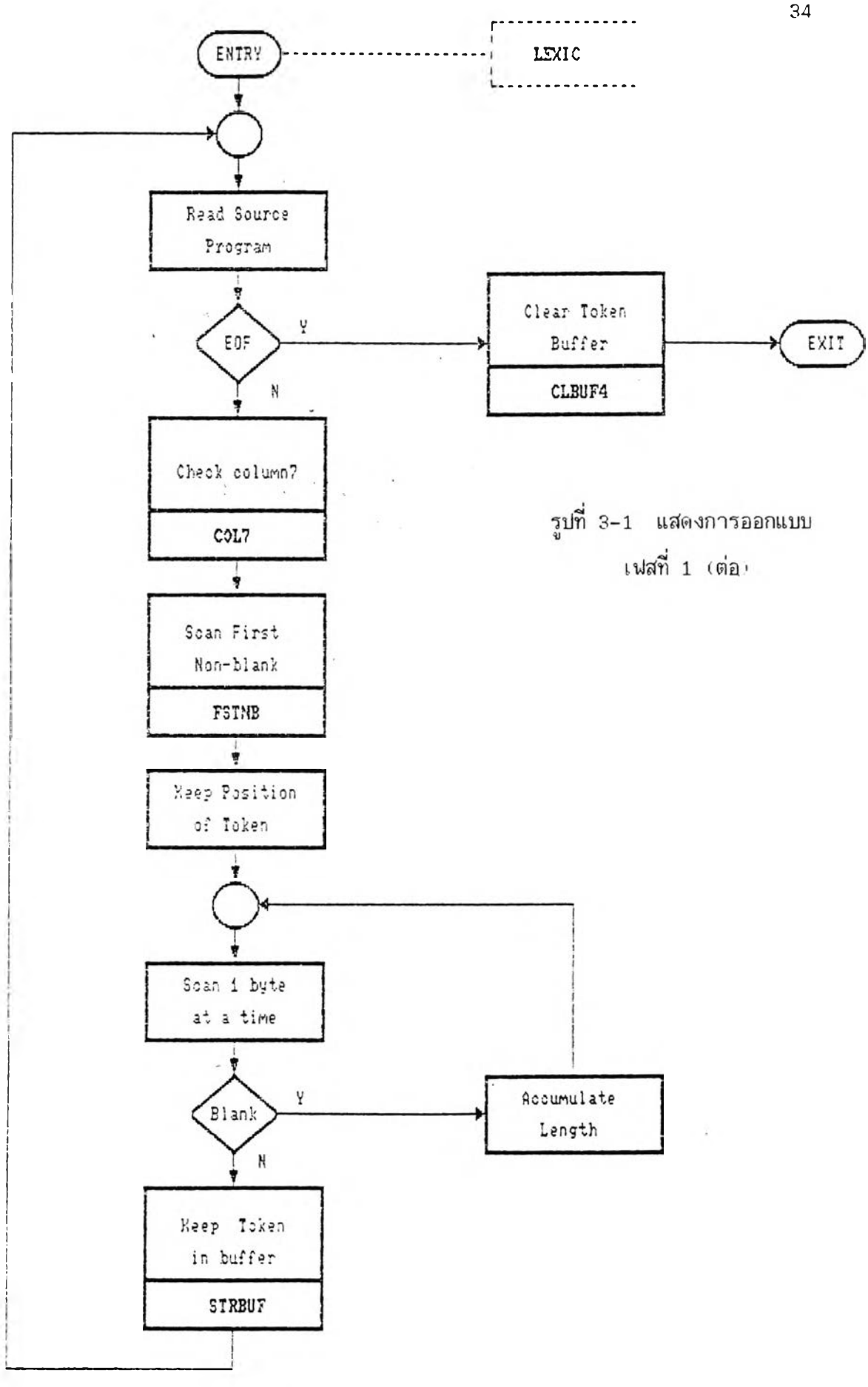
<u>ข้อมูลที่จำเป็นในการสร้างเฟสการวิเคราะห์เลกซิเคิล</u>

ในการออกแบบชั้นที่ 2 ของเฟสที่ 1 คือการกำหนดโครงสร้างของข้อมูล (Data Structure) ที่จำเป็นต้องใช้ในการพัฒนาตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล ได้แก่

1. โปรแกรมดิบภาษาโคบอลที่นำเข้าสู่การประมวลผล
2. ตัวนับบรรทัด (Line Counter - COUNT1) ใช้ในการติดตามหมายเลขที่กำหนดให้แต่ละบรรทัดของโปรแกรมดิบ เพื่อนำไปอ้างอิงในการสร้างรหัสแสดงความผิดพลาด
3. ตัวเก็บรหัสความผิดพลาด (ERRCD) เป็นเนื้อที่ในหน่วยความจำที่ใช้เก็บบันทึกรหัสความผิดพลาดที่เกิดในขณะนั้น
4. บัฟเฟอร์ ในหน่วยความจำที่ใช้รับข้อมูลโปรแกรมดิบที่อ่านเข้า (IBMBUF)
5. เนื้อที่ใช้งาน (Work Area) ในการตรวจสอบความถูกต้อง และสร้างโทเกน (DMA)
6. บัฟเฟอร์ของแฟ้มข้อมูลที่เก็บรหัสความผิดพลาดจากเฟสที่ 1 (BUFFER)
7. บัฟเฟอร์ของแฟ้มข้อมูลที่ลอกมาจากบัฟเฟอร์ข้อมูลนำเข้า เพื่อสร้างเลขลำดับของบรรทัดและบันทึกข้อมูลโปรแกรมดิบที่คัดลอกมาเตรียมบันทึกเก็บ ใช้เนื้อที่



รูปที่ 3-1 แสดงการออกแบบ
เฟสที่ 1



รูปที่ 3-1 แสดงการออกแบบเฟสที่ 1 (ต่อ)

เดียวกันกับเนื้อที่ใช้งาน (DMA)

8. บัฟเฟอร์ของแฟ้มข้อมูลโทเกน (LEXBUF)
9. ตัวเก็บคอลัมน์ที่เริ่มต้นของโทเกนเพื่อใช้เก็บในแฟ้มข้อมูลโทเกน ไว้ใช้ในการตรวจสอบต่อไป (COLUMN)
10. ตัวเก็บความยาวของโทเกน (LENGTH)

รูปแบบและโครงสร้างข้อมูลที่ใช้

รูปแบบและโครงสร้างของข้อมูลต่าง ๆ ที่จำเป็นในการวิเคราะห์เลกซิเคิล ดังได้กล่าวมาแล้วนั้น สามารถสร้างขึ้นโดยระบุรูปแบบที่แน่นอนได้ ดังต่อไปนี้

โปรแกรมดิบภาษาโคบอล

ดังได้กล่าวมาแล้วว่าโปรแกรมดิบภาษาโคบอลที่จะถูกนำมาตรวจสอบนั้น ถูกสร้างขึ้นโดยซอฟต์แวร์ระบบเตรียมข้อมูล "ไทยทำ" โดยถูกจัดเก็บในแผ่นจานแม่เหล็กแบบ IBM-3470 กล่าวคือ รูปแบบดังกล่าวจะมีลักษณะสำคัญ ดังนี้

1. ประกอบด้วย 26 เซกเตอร์ 77 แทรค
2. การเก็บข้อมูลจะเก็บ 1 เซกเตอร์ต่อ 1 ระเบียน (Sector Boundary) โดยมีความยาวคงที่ระเบียนละ 128 ไบต์ ซึ่งระเบียนในแต่ละแฟ้มข้อมูลจะต้องจัดอยู่ในลักษณะเรียงชิดติดกันจริงตามกายภาพ (Physical Contiguous)

จะเห็นได้ว่ารูปแบบดังกล่าวนั้น ทำให้การเก็บโปรแกรมดิบภาษาโคบอลแต่ละบรรทัด ซึ่งมีเพียง 80 ไบต์ จะต้องถูกจัดเก็บโดยใช้เนื้อที่ 128 ไบต์เสมอกล่าวคือจะต้องทิ้ง 48 ไบต์ที่เหลือทิ้งไปโดยไม่สามารถใช้เนื้อที่นั้นได้

ใช้งาน 80 ไบต์	เหลือไม่ใช้งาน 48 ไบต์

รหัสที่เก็บโปรแกรมดิบภาษาโคบอลเป็นรหัส EBCDIC และไม่มีข้อมูลเพื่อการควบคุม (Control Information) ใด ๆ ประกอบอยู่ภายในระเบียนข้อมูลนั้น

ลักษณะคำสั่งแต่ละบรรทัดของภาษาโคบอลนั้น อาจจะแบ่งได้ดังตัวอย่าง

```
0.....1.....2.....3.....4.....7.....8
1.....78...2.....23.....0
000080 ENVIRONMENT DIVISION.                              TEST0010
000090 CONFIGURATION SECTION.                             TEST0010
000100 SOURCE-COMPUTER.  IBM-3031.                        TEST0010
000110 OBJECT-COMPUTER.  IBM-3031.                        TEST0010
000120 INPUT-OUTPUT SECTION.                              TEST0010
000130 FILE-CONTROL.                                      TEST0010
000140     SELECT INPUT-FILE ASSIGN TO UT-S-SYSIN.        TEST0010
000150     SELECT OUT-FILE ASSIGN TO UT-S-SYSPRINT.       TEST0010
```

จากตัวอย่างนี้ สามารถอธิบายความหมายของแต่ละส่วนได้ คือ

1. คอลัมน์ 1-6 เป็นเลขลำดับบรรทัด (Serial Number)
2. คอลัมน์ 7 เพื่อใส่เครื่องหมายบรรทัดหมายเหตุ, ต่อบัตร และอื่น ๆ
3. คอลัมน์ 8-11 เป็นส่วน A (Area A)
4. คอลัมน์ 12-72 เป็นส่วน B (Area B)
5. คอลัมน์ 73-80 บอกชื่อของโปรแกรม (Identification)

ลักษณะแฟ้มข้อมูลโทเกน (CBLSK4.TKN)

แฟ้มข้อมูลโทเกน เป็นแฟ้มข้อมูลที่เก็บโทเกนที่ได้จากการวิเคราะห์เลกซิเคิล ในเฟสที่ 1 เพื่อนำไปใช้ต่อในเฟสที่ 2 มีลักษณะดังนี้

1. แต่ละระเบียนยาว 128 ไบต์
2. เก็บในแผ่นจานแม่เหล็กภายใต้ระบบ ซีพีเอ็ม
3. แต่ละระเบียนประกอบด้วยเนื้อหาดังนี้

สมาชิกที่ 1	สมาชิกที่ 2	สมาชิกที่ 3	.....

สมาชิกแต่ละตัวมีความยาวไม่เท่ากัน ขึ้นอยู่กับความยาวของโทเกนและประเภทของโทเกนซึ่งในแต่ละสมาชิกมีส่วนประกอบดังนี้

3.1 ไบต์แรกนำหน้า ถ้าเป็นค่า 00H หมายถึงอีก 4 ไบต์ที่ตามมา เป็นตัวเลขบอกเลขหมายบรรทัดที่สร้างขึ้น เพื่ออ้างอิง (COUNT1)

3.2 ส่วนที่ถัดมาของโทเกน มีลักษณะดังนี้

3.2.1 ถ้าไบต์แรกไม่ใช่ 00H จะหมายถึงว่า ไบต์นั้นคือคอลัมน์เริ่มต้นของโทเกนที่จะตามมา (แสดงในลักษณะเลขฐานสอง)

3.2.2 ไบต์ถัดมา 1 ไบต์ แสดงถึงความยาวของโทเกนที่ตามมา

3.2.3 ส่วนที่ตามมาเป็นเนื้อหาของโทเกน ยาวเท่ากับค่าที่ระบุในไบต์ที่บอกความยาว

00H	30H	30H	30H	31H	08H	0EH	I	D	E	N	T	I	F	I

C	A	T	I	O	N	17H	08H	D	I	V	I	S	.....

3.3 แต่ละสมาชิกจะถูกบันทึกเรียงต่อกันไปเรื่อย ๆ จนกว่าจะเต็มบัฟเฟอร์ 128 ไบต์ หรือเหลือที่ในบัฟเฟอร์ไม่พอ ถ้าเหลือที่ในบัฟเฟอร์ไม่พอจะถูกเติม (Pad) ค่า OFFH ให้จนเต็ม 128 ไบต์

<u>ลักษณะแฟ้มข้อมูลรหัสความผิดพลาด</u> (CBLSK3.ERR)

เป็นข้อมูลที่เก็บรหัสความผิดพลาดจากการตรวจพบในเฟสที่ 1 เพื่อนำรหัสดังกล่าวไปแปลงเป็นข้อความบอกความผิดในเฟสที่ 3 มีลักษณะดังนี้

1. ความยาวของระเบียนแต่ละระเบียนยาว 128 ไบต์
2. เก็บในแผ่นจานแม่เหล็กภายใต้ระบบซีพีเอ็ม
3. 1 ระเบียนประกอบด้วย 16 สมาชิก ในแต่ละสมาชิกมีลักษณะดังนี้

b	b	C_1	C_2	L_1	L_2	L_3	L_4

- bb เป็นช่องว่าง 2 ช่อง
- C_1C_2 รหัสความผิดพลาดยาว 2 byte
- L_1 - L_4 เป็นเลขอ้างอิงของบรรทัดที่ผิด

(ตัวอย่างแสดงในรูปที่ 3-2)

```
....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

20200032303031322A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A     .20012****************
2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A  ************************
2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A  ************************
2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A  ************************
2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A  ************************
2A2A2A2A2A2A2A2A                                    ********
```

รูปที่ 3-2 ก. แสดงการเก็บข้อมูลความผิดพลาดในแฟ้มข้อมูล

CBLSK3.ERR และ CBLSK5.ERR

```
....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

0030303031080E4944454E54494D4941544494F4E170844   .0001..IDENTIFICATION..D
49564953494F4E1F012E00303030320B0A50524F47524144   IVISION....0002..PROGRAM
2D494412012E140854484553495330341C012E0030303033   -ID.....THESIS04....0003
080752454D41524B530F012E140556414C49441A0750524F   ..REMARKS.....VALID..PRO
4752414D2202544F250453484F572A0C5645524946494341   GRAM".TO%.SHOW*.VERIFICA
54494F4EFFFFFFFFF                                  TION....

36012E00303030350808454E5649524F4E4D454E54140844   6...0005..ENVIRONMENT..D
49564953494F4E1C012E0030303036080D434F4E46494755   IVISION....0006..CONFIGU
524154494F4E160753454354494F4E1D012E003030303708   RATION..SECTION....0007.
0F534F555243452D434F4D505554455217012E190849424D   .SOURCE-COMPUTER.....IBM
2D333033312101 2E0030303038080F4F424A4543542D434F   -3031!...0008..OBJECT-CO
4D5055544552FFFF                                   MPUTER..

17012E190849424D2D333033312101 2E0030303039080C49   .....IBM-3031!...0009..I
4E5055542D4F5554505554150753454354494F4E1C012E00   NPUT-OUTPUT..SECTION....
3030313008 0C46494C452D434F4E54524F4C14012E003030   0010..FILE-CONTROL....00
31310C0653454C454354130A494E5055542D46494C451E06   11..SELECT..INPUT-FILE..
41535349474E2502544F280A55542D532D535953494E3201   ASSIGN%.TO(.UT-S-SYSIN2.
2EFFFFFFFFFFFFFF                                   ........

00303031320C0653454C454354130B4F55545055542D4649   .0012..SELECT..OUTPUT-FI
4C451F0641535349474E2602544F290B55542D5359535052   LE..ASSIGN&.TO).UT-SYSPR
494E5434012E0030303134080444415441 1D084449564953   INT4...0014..DATA..DIVIS
494F4E15012E0030303135080446494C450D075345435449   ION....0015..FILE..SECTI
4F4E14012E0030303136080246440C0A494E5055542D4649   ON....0016..FD..INPUT-FI
4C45FFFFFFFFFFFF                                   LE......

18054C4142454C1E065245434F52442502495328074F4D49   ..LABEL..RECORD%.IS(.OMI
5454454400303031371804444154411D065245434F524424   TTED.0017..DATA..RECORD$
024953270C494E5055542D5245434F524433012E00303031   .IS'.INPUT-RECORD3...001
380802303 10C0C494E5055542D5245434F52442303504943   8..01..INPUT-RECORD#.PIC
27055828383029 2C012E0030303230080246 44FFFFFFFFF   '.X(80),...0020..FD.....
FFFFFFFFFFFFFFFF                                   ........

0C0B4F55545055542D46494C451C054C4142454C22065245   ..OUTPUT-FILE..LABEL".RE
434F52442902495 32C074F4D4954544544003030323 11C04   CORD).IS,.OMITTED.0021..
44415441210652454 34F52442802495 32B0D4F5554505554   DATA!.RECORD(.IS+.OUTPUT
2D5245434F524438012E0030303232080230310C0D4F5554   -RECORD8...0022..01..OUT
5055542D5245434F52442303504943270658283133332292D   PUT-RECORD#.PIC'.X(132)-
012EFFFFFFFFFFFF                                   ........

0030303234080F574F524B494E472D53544F524147451807   .0024..WORKING-STORAGE..
53454354494F4E1F012E0030303235080230310C1150524F   SECTION....0025..01..PRO
4752414D2D5641524941424C45531D012E00303032360C02   GRAM-VARIABLES....0026..
3033100A5641524941424C452D312303504943270658283 1   03..VARIABLE-1#.PIC'.X(1
3332292D012E00303032370C023033100A5641524941424C   32)-...0027..03..VARIABL
452D32FFFFFFFFFF                                   E-2.....

230350494327053928303329 2D0556414C554533045A4552   #.PIC'.9(03)-.VALUE3.ZER
4F37012E00303032380C023033100B5641524941424C452D   07...0028..03..VARIABLE-
3232230350494327053928303329 2D0556414C554533045A   22#.PIC'.9(03)-.VALUE3.Z
45524F37012E00303032390C023033100A5641524941424C   ERO7...0029..03..VARIABL
452D331D064F4343555253240133260554494D45532B012E   E-3..OCCURS$.3&.TIMES+..
0030303330FFFFFF                                   .0030...
```

```
....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

10023035130A5641524941424C452D3431F064F4343555253  ..05..VARIABLE-4..OCCURS
26013328055449 4D4553300350494334055828313029390 1 &.3(.TIMES0.PIC4.X(10)9.
2E00303033320809505 24F43454455524512084449564953  ..0032..PROCEDURE..DIVIS
494F4E1A012E00303033330 80C4D41494E2D524F5554494E  ION....0033..MAIN-ROUTIN
4514012E00303033340C044F50454E1105494E505554FFFF  E....0034..OPEN..INPUT..
FFFFFFFFFFFFFFFF                                  ........

170A494E5055542D46494C4500303033351 1064F55545055  ..INPUT-FILE.0035..OUTPU
54180B4F55545055542D46494C4523012E00303033360C04  T..OUTPUT-FILE#...0036..
4D4F5645110553504143451702544F1A0A5641524941424C  MOVE..SPACE..TO..VARIABL
452D3124012E00303033370C0452454144110A494E505554  E-1$...0037..READ..INPUT
2D46494C451C0241541F03454E4423044D4F56452803414C  -FILE..AT..END#.MOVE(.AL
4C2C03275827FFFF                                  L,.'X'..

3002544F330A5641524941424C452D313D012E0030303338  0.TO3.VARIABLE-1=...0038
0C07504552464F524D140A4D4F56452D53504143451F0756  ..PERFORM..MOVE-SPACE..V
415259494E47270A5641524941424C452D32320446524F4D  ARYING'.VARIABLE-22.FROM
37013139024259 3C013100303033391405554E54494C1A0A  7.19.BY<.1.0039..UNTIL..
5641524941424C452D3225024953280747524541544552 30  VARIABLE-2%.IS(.GREATER0
045448414EFFFFFF                                  .THAN...

350133003030343014054146544552 1A0B5641524941424C  5.3.0040..AFTER..VARIABL
452D3232260446524F4D2B01312D024259300131 00303034  E-22&.FROM+.1-.BY0.1.004
311405554E54494C1A0B5641524941424C452D3232260249  1..UNTIL..VARIABLE-22&.I
53290747524541544552310454484 14E36013337012E0030  S).GREATER1.THAN6.37...0
3034320C055752495445120D4F55545055542D5245434F52  042..WRITE..OUTPUT-RECOR
44200446524F4DFF                                  D .FROM.

250A5641524941424C452D313005414654455236094144 56  %.VARIABLE-10.AFTER6.ADV
414E43494E474001324205 4C494E455347012E0030303433  ANCING@.2B.LINESG...0043
0C05434C4F5345120A494E5055542D464941451E0B4F5554  ..CLOSE..INPUT-FILE..OUT
5055542D46494C4529012E00303034340C0453544F501103  PUT-FILE)...0044..STOP..
52554E14012E00303034360 80A4D4F56452D535041434512  RUN....0046..MOVE-SPACE.
012EFFFFFFFFFFFF                                  ........

00303034370C044D4F5645110553504143451702544F1A0A  .0047..MOVE..SPACE..TO..
5641524941424C452D34250B285641524941424C452D3230  VARIABLE-4%.(VARIABLE-20
012C320C5641524941424C452D3232293E012EFFFFFFFFFF  .,2.VARIABLE-22)>.......
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  ........................
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  ........................
FFFFFFFFFFFFFFFF                                  ........
```

รูปที่ 3-2 ข. แสดงการเก็บข้อมูลโทเกนในแฟ้มข้อมูล CBLSK4.TKN

ถ้าในระเบียนนั้นไม่ครบ 16 สมาชิก ที่ว่างที่เหลือจะเติมตัวอักษร "*" จนเต็ม 128 ไบต์

ลักษณะแฟ้มข้อมูลที่ลอกมาจากแฟ้มตั้งต้นของ IBM (CBLSK1.PRN)

CBLSK1.PRN เป็นแฟ้มข้อมูล ที่คัดลอกเนื้อหาที่อ่านได้จากโปรแกรมดิบในแผ่นจานแม่เหล็กของ IBM-3470 แล้วนำมาสร้างเลขอ้างอิงบรรทัด (SEQGEN) จากนั้นจึงคัดลอกลงเก็บบันทึกในแผ่นจานแม่เหล็กภายใต้ระบบซีพีเอ็ม เพื่อนำไปประกอบในการสร้างข่าวสารความผิดพลาดจากการตรวจสอบในเฟสที่ 3 มีลักษณะดังนี้

1. ความยาวระเบียนยาว 128 ไบต์
2. เนื้อหาจริงเก็บเพียง 80 ไบต์แรก เหลือ 48 ไบต์ว่างไว้
3. ลักษณะของแต่ละระเบียน เหมือนกับลักษณะของการเขียนโปรแกรมดิบภาษาโคบอลทุกประการ
4. เก็บโดยรหัส ASCII

เนื้อที่ใช้งานตรวจสอบความถูกต้องและสร้างโทเกน

ใช้ส่วนของหน่วยความจำภายใต้ระบบซีพีเอ็ม ตั้งแต่แอดเดรส 00H 80H ถึง 01H 00H เป็นจำนวน 128 ไบต์ เรียกว่า ดีเอ็มเอ (DMA-Data Management Area) โดยระบบในเฟสที่ 1 จะใช้เนื้อที่บริเวณนี้เป็นเนื้อที่ทำงานแต่ไม่มีการเปลี่ยนแปลงเนื้อหาใด ๆ ที่ได้คัดลอกมาจาก IBMBUF

ตัวเก็บค่าและติดตามที่จำเป็นอื่น ๆ

ในเฟสที่ 1 ยังมีการจัดเนื้อที่หน่วยความจำหลัก เพื่อใช้เก็บค่าต่าง ๆ ที่ต้องการติดตาม ได้แก่

1. ตัวนับบรรทัด ใช้เป็นตัวติดตามบรรทัดปัจจุบัน เกิดจากการนับสะสมบรรทัดในโปรแกรม SEQGEN มีชื่อ COUNT1 ยาว 4 ไบต์ ค่าเริ่มแรกเป็น 00H 00H 00H 00H จะถูกเก็บค่าในรหัส ASCII ที่เป็นรหัสแสดงได้ กล่าวคือ ตัวเลข 0001 ในการแสดงจะถูกเก็บเป็น 30H 30H 30H 31H
2. ตัวเก็บรหัสความผิดพลาด ใช้เป็นตัวเก็บรหัสแสดงความผิดที่พบ โดยเก็บไว้ชั่วคราวก่อนจะย้ายไปลงบัฟเฟอร์ของ CBLSK3.ERR ต่อไป มีชื่อว่า ERRCD มีความยาว 2 ไบต์
3. ตัวเก็บคอลัมน์เริ่มต้นของโทเกน ใช้เก็บคอลัมน์เริ่มต้นของโทเกนที่พบ ก่อนที่จะย้ายไปลงบัฟเฟอร์ของ CBLSK4.TKN มีชื่อ COLUMN มีความยาว 1 ไบต์ เก็บในลักษณะเลขฐานสอง
4. ตัวเก็บความยาวโทเกน ใช้นับความยาวของโทเกน เพื่อจะเก็บเป็นข่าวสารควบคุมในแฟ้ม CBLSK4.TKN มีความยาว 1 ไบต์ เก็บในลักษณะเลขฐานสอง

วิธีการและอัลกอริธึมในเฟสที่ 1

ในเฟสที่ 1 นี้ทำหน้าที่หลัก ๆ คือ เพื่อสร้างโทเกนจากโปรแกรมติบ เพื่อนำไปใช้วิเคราะห์ด้วยพาสเซอร์ในเฟสที่ 2 หน้าที่ต่าง ๆ ในเฟสที่ 1 แสดงได้ดังรูป 3-3

การตรวจสอบคอลัมน์ 7

ในคอลัมน์ 7 ของโปรแกรมภาษาโคบอล จะมีตัวอักษรที่เป็นไปได้คือ ช่องว่าง, *, /, D, - เท่านั้น โดยแต่ละตัวก็มีความหมายต่าง ๆ กันดังได้กล่าวแล้ว ถ้าหากพบว่าในคอลัมน์ดังกล่าวไม่เป็นตัวอักษรที่ถูกต้อง ก็จะให้รหัสความผิดพลาด 00H 01H อันมีความหมายว่ามีตัวอักษรที่ผิดพลาดถูกตรวจพบในคอลัมน์ดังกล่าว (Invalid character in column 7) แล้วโปรแกรมย่อย ERROR จะเป็นผู้จัดการเก็บรหัสความผิดพลาดนั้นกับเลขประจำบรรทัด (COUNT1) ลงใน CBLSK3.ERR

การสร้างโทเกนจากโปรแกรมดิบ

ในภาษาโคบอลนั้นจะมีตัวอักษรที่สามารถทำให้รับรู้ได้ว่าสายวลีที่ผ่านจากการรับทีละ 1 ตัวอักษรรวมเข้าด้วยกันนั้นสิ้นสุดซึ่งถือว่าเป็น 1 โทเกน โดยตัวอักษรต่าง ๆ ดังกล่าวนั้นถือเป็นตัวบอกจุดสิ้นสุดคำ (Delimiter) ได้แก่

1. ช่องว่าง
2. จุด (Period) ที่ไม่ใช่ส่วนหนึ่งของค่าคงที่ตัวเลข
3. เครื่องหมายวรรคตอน ได้แก่
   - จุลภาค (, Comma)
   - มหัพภาค (; Semi-colon)
   - อัญประกาศ (' Quote)
4. เครื่องหมายทางคณิตศาสตร์ที่ไม่ใช่บอกเครื่องหมายของค่าคงที่ตัวเลข (ไม่เป็น Unany sign) ได้แก่
   - +, -, *, /, **
   - เครื่องหมายสมการ =
   - เครื่องหมายวงเล็บ ( )

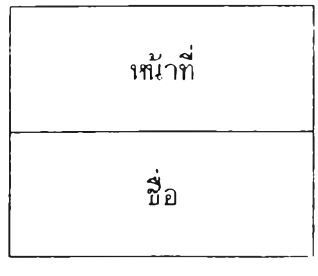
เครื่องหมายต่าง ๆ ดังกล่าวนี้ ตัววิเคราะห์เลกซิเคิลจะรับรู้เป็นโทเกน 1 โทเกน

การสร้างโทเกนพร้อมกับการตรวจหาที่ผิดในเฟสนี้ทำได้โดยอ่านโปรแกรมดิบเข้ามาทีละ 1 ระเบียนซึ่งมี 1 บรรทัด เนื่องจากแผ่นจานแม่เหล็กของระบบ IBM-3470 เก็บบันทึกในลักษณะดังกล่าว แล้วเริ่มตรวจตั้งแต่คอลัมน์ที่ 8 ไป โดยใน 7 คอลัมน์แรกส่วนของการตรวจสอบคอลัมน์ 7 จะทำการตรวจสอบแล้ว ตั้งแต่คอลัมน์ที่ 8 นั้นจะมีส่วนทำงานย่อย (Routine) ชื่อว่า FSTNB ตรวจหาตัวอักษรตำแหน่งแรกที่ไม่ใช่ช่องว่าง (Non-blank Character) แล้วส่งตำแหน่งดังกล่าวกลับมายังส่วนทำงานหลัก (Main Routine) จากนั้นส่วนทำงานหลักจะเก็บบันทึกตำแหน่งนั้น คือ เป็นคอลัมน์ที่โทเกนเริ่มต้นลงในตัวแปร COLUMN

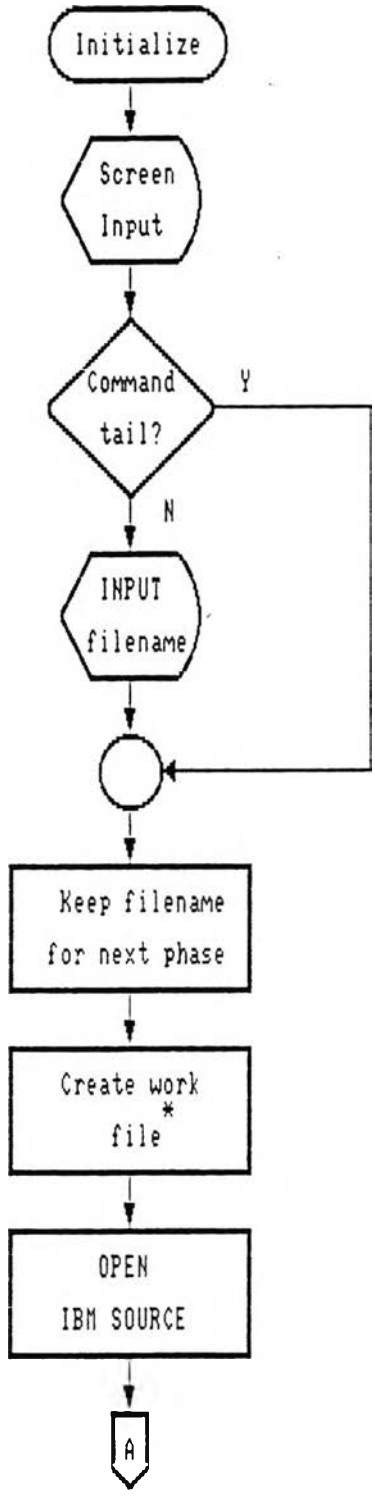
เพื่อเก็บในบัฟเฟอร์ จากนั้นก็อ่าน (Scan) ไปทีละตัวอักษรพร้อมกับบวกสะสมความยาวลงในตัวแปรชื่อว่า LENGTH ด้วย จนกว่าจะอ่านพบตัวอักษรบอกจุดสิ้นสุดจึงจะถือว่าโทเกนนั้นจบลง จากนั้นจะเรียกส่วนทำงานย่อยชื่อ STRBUF เก็บโทเกนนั้นลงบัฟเฟอร์ซึ่งจะเก็บ COLUMN, LENGTH และเนื้อหาของโทเกนตามลำดับ ส่วนทำงานย่อย STRBUF จะเรียกส่วนทำงานย่อยที่ชื่อ CHKBUF ตรวจดูบัฟเฟอร์ว่าเต็ม 128 ไบต์ หรือมีที่พอเพียงในการเก็บโทเกนนั้นหรือไม่ ถ้าพอจะเก็บลงในบัฟเฟอร์ ถ้าไม่พอจะถ่ายบัฟเฟอร์นั้นไปเก็บในแผ่นจานแม่เหล็กก่อนแล้วจึงนำโทเกนใหม่ลงเก็บ ทำกระบวนการเช่นนี้จนกว่าจะหมดแฟ้มข้อมูลโปรแกรมดิบ จะเรียกส่วนทำงานย่อยชื่อ CLBUF4 จัดการบัฟเฟอร์ครั้งสุดท้ายบันทึกในแผ่นจานแม่เหล็ก และเรียกส่วนทำงานย่อยชื่อ CLBUF3 จัดการบัฟเฟอร์แฟ้มข้อมูลเก็บรหัสความผิดพลาดบันทึกในแผ่นจานแม่เหล็ก

การแบ่งส่วนย่อยในเฟสที่ 1

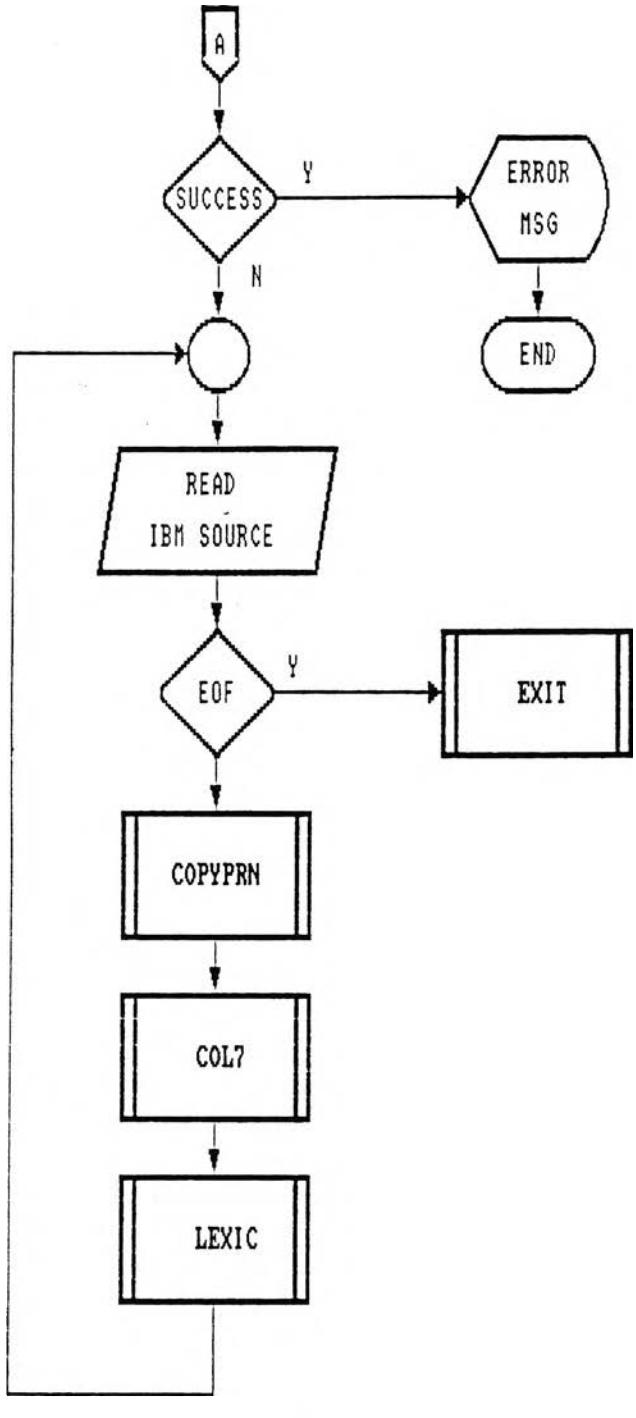
ในภาษาลำดับงานที่แสดงนั้น ใช้รูปแบบเพื่อบอกถึงส่วนการทำงานย่อย แต่ละส่วนทั้งที่เขียนส่วนนั้นในลักษณะเรียกใช้แล้วกลับมาทำต่อในส่วนหลัก (CALL) และเขียนในลักษณะข้ามไปทำงานในส่วนนั้น ๆ ต่อ (JUMP) โดยสามารถแสดงรายละเอียดการแบ่งส่วนย่อยในเฟสที่ 1 ได้ดังนี้



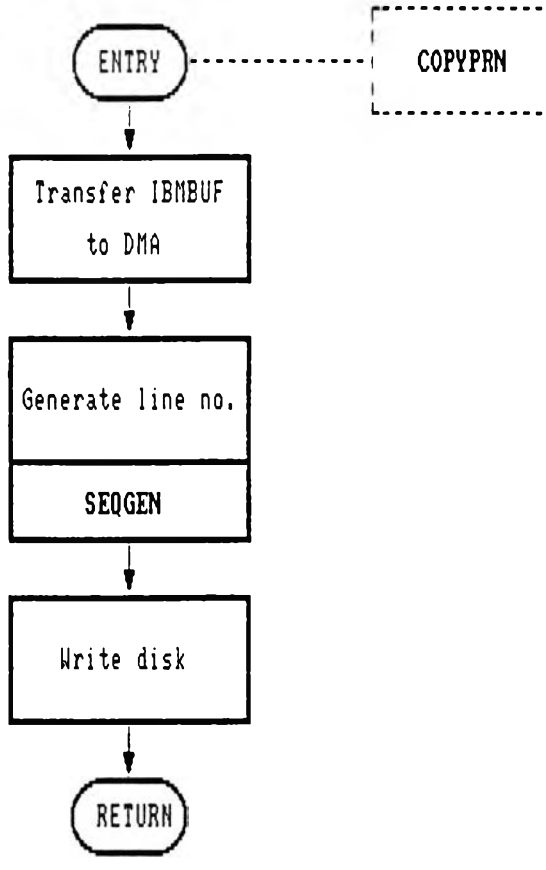
1. SCAN เป็นส่วนหลักใหญ่ควบคุมการทำงานส่วนย่อยทั้งหมด
2. PHASE0 เป็นส่วนทำงานเริ่มแรก (Initialize) โดยตรวจหาแฟ้มข้อมูลที่จำเป็นในสารบัญของแผ่นจานแม่เหล็ก (Directory) ถ้าพบจะลบออกและสร้างใหม่ ได้แก่ แฟ้ม CBLSK1.PRN, CBLSK3.ERR และ CBLSK4.TKN



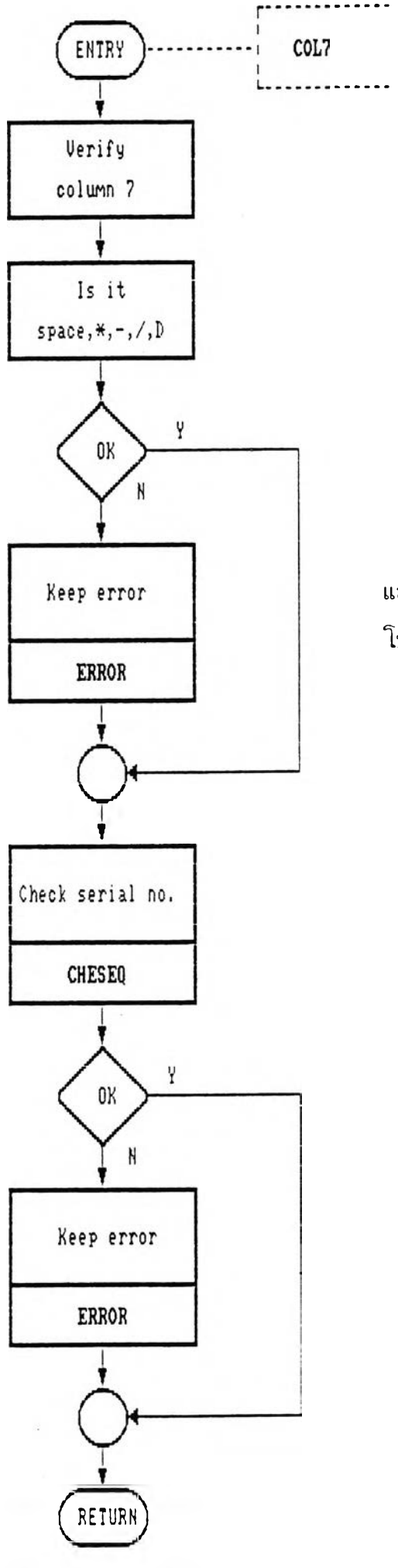
รูปที่ 3-3 แสดงอัลกอริธึมของเฟสการวิเคราะห์เลกซิเคิล



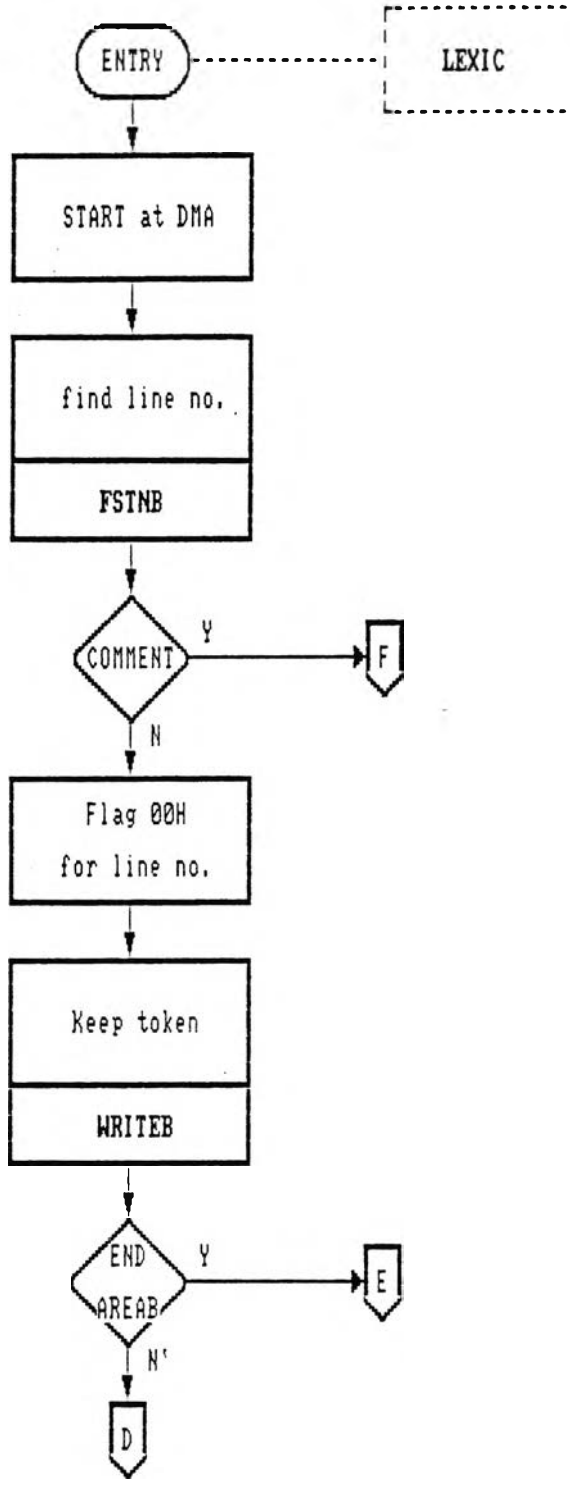
รูปที่ 3-3 แสดงอัลกอริธึมในการออกแบบเฟสการวิเคราะห์เลกซิเคิล (ต่อ)



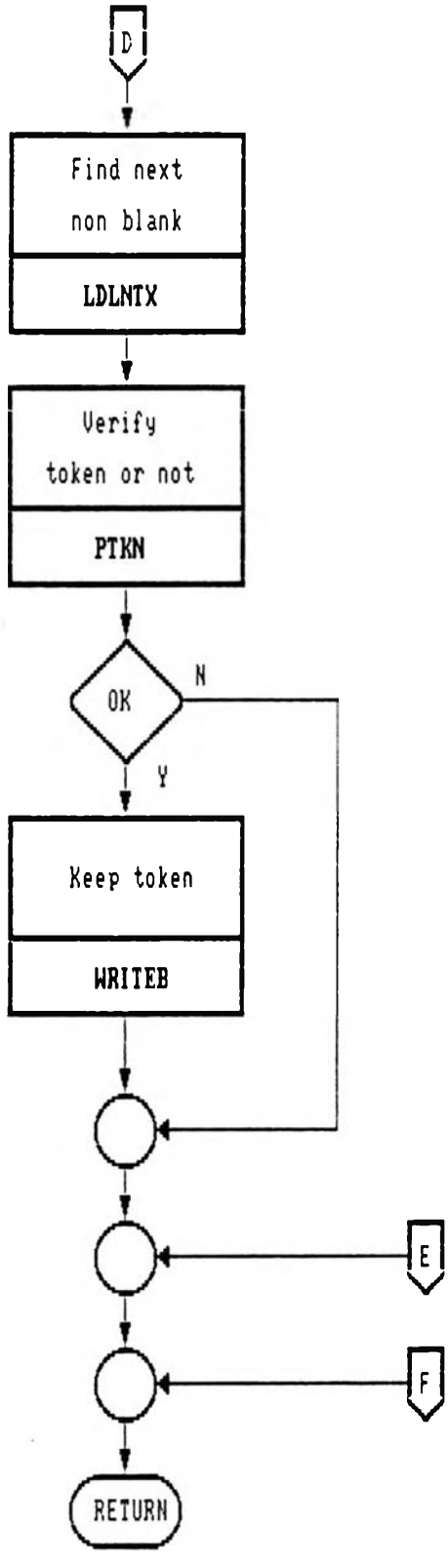
รูปที่ 3-4 แสดงอัลกอริธึมของโปรแกรม COPYPRN



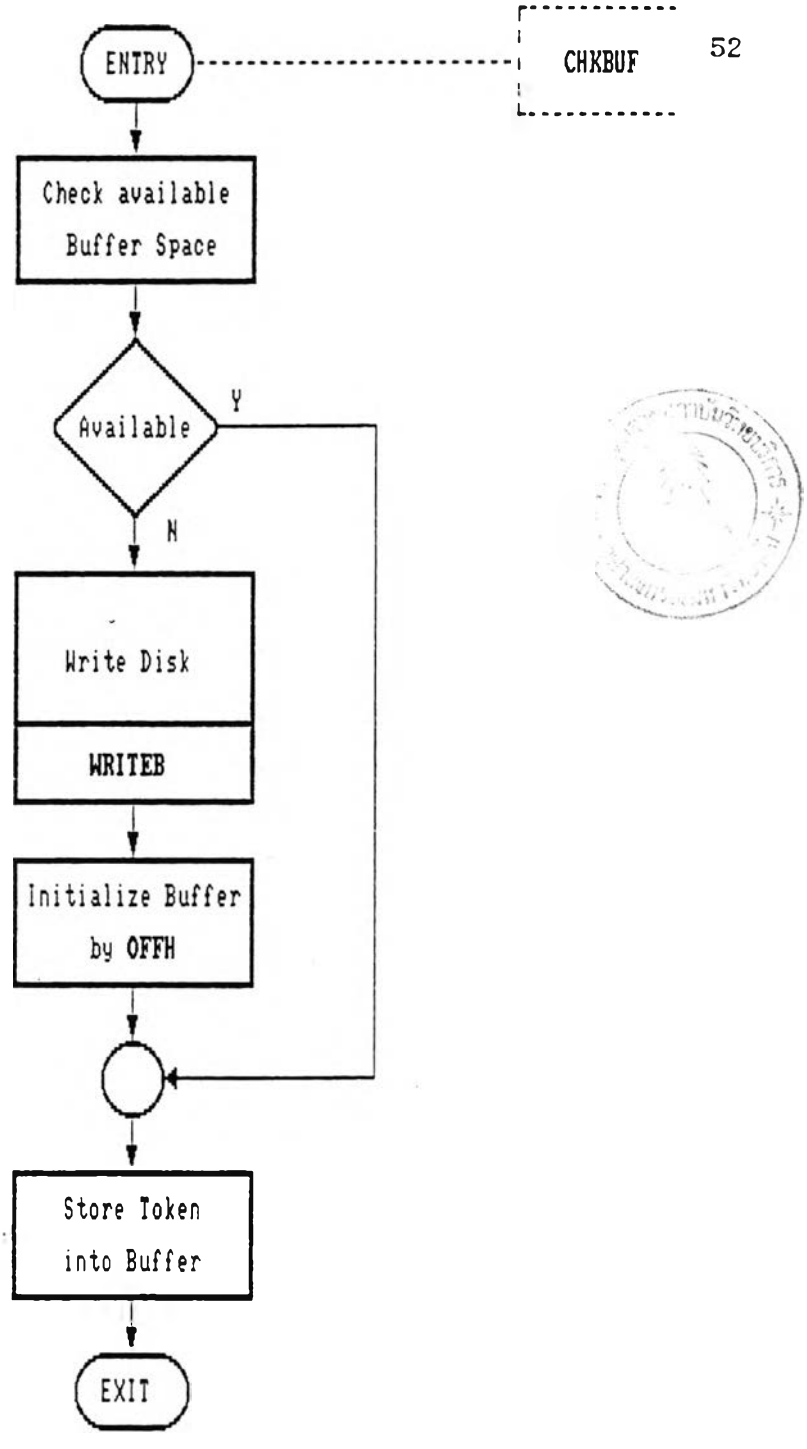
รูปที่ 3-5
แสดงอัลกอริธึมของ
โปรแกรม COL7



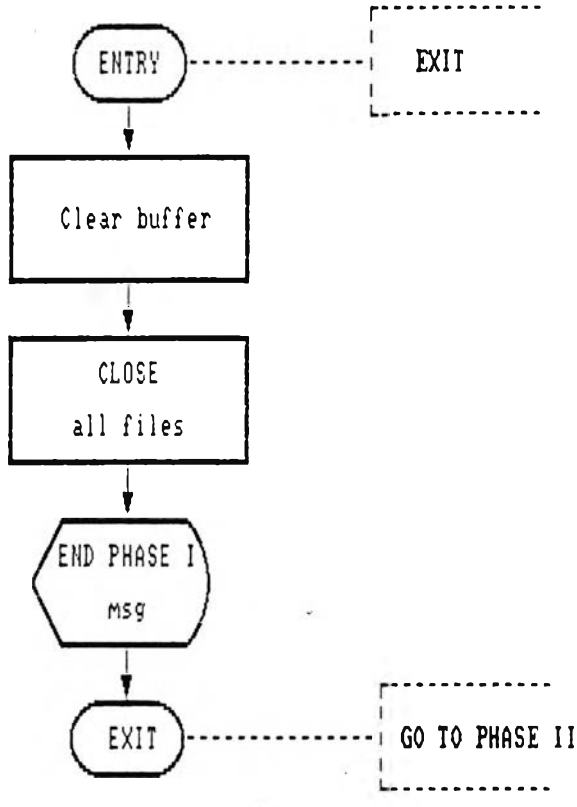
รูปที่ 3-6 แสดงอัลกอริธึมของโปรแกรม LEXIC



รูปที่ 3-6 แสดงอัลกอริธึมของโปรแกรม LEXIC (ต่อ)



รูปที่ 3-7 แสดงอัลกอริธึมของโปรแกรม CHKBUF



รูปที่ 3-8 แสดงอัลกอริธึมของโปรแกรม EXIT

3. READ1 ทำหน้าที่อ่านแฟ้มข้อมูลโปรแกรมดิบ IBM-3470 เข้าสู่ระบบ
4. COPYPRN ส่วนทำงานย่อยที่ลอกข้อมูลโปรแกรมดิบลงในแฟ้ม CBLSK1.PRN
5. SEQGEN ส่วนทำงานย่อยที่สร้างเลขลำดับคำสั่งของโปรแกรมดิบแต่ละบรรทัด
6. COL7 ส่วนทำงานย่อยทำหน้าที่ตรวจสอบคอลัมน์ 7 ของคำสั่งแต่ละบรรทัด และตรวจคอลัมน์ 1-6 ว่ามีเลขลำดับที่ผู้ใช้เขียนขึ้นเรียงลำดับถูกต้องหรือไม่
7. ERROR เป็นส่วนทำงานย่อย ที่ทำหน้าที่เก็บบันทึกรหัสความผิดพลาดลงในแฟ้ม CBLSK3.ERR
8. LEXIC ส่วนทำงานย่อยสร้างโทเกนและตรวจสอบความผิดที่อาจจะพบได้

เฟสที่ 2 การวิเคราะห์ความถูกต้องของไวยากรณ์

นิยามของปัญหา

ในเฟสที่ 2 เป็นการรับข้อมูลจากเฟสที่ 1 ที่ได้จากการวิเคราะห์เลกซิเคิล โดยข้อมูลที่รับมาจะอยู่ในแฟ้มข้อมูล CBLSK4.TKN โดยในเฟสนี้ อาจจะสามารถแบ่งแยกหน้าที่หลักออกเป็น 4 ส่วน ได้แก่

1. ตรวจสอบความถูกต้องในส่วน IDENTIFICATION DIVISION
2. ตรวจสอบความถูกต้องในส่วน ENVIRONMENT DIVISION
   2.1 เก็บชื่อนิเศษที่ใช้ควบคุมเครื่องพิมพ์ (SPECIAL-NAMES)
   2.2 เก็บชื่อแฟ้มข้อมูลในวลี SELECT พร้อมทั้งลักษณะว่าเป็นแฟ้มข้อมูลนำเข้าหรือออก ขึ้นอยู่กับว่าเป็น UT-S-SYSIN หรือ UT-S-SYSPRINT
3. ตรวจสอบความถูกต้องในส่วน DATA DIVISION
   3.1 สร้างตารางสัญญลักษณ์ (Symbol Table) เพื่อเก็บตัวแปรต่าง ๆ ไปตรวจสอบต่อไป
   3.2 ตรวจไวยากรณ์การกำหนดตัวแปรแต่ละตัว
4. ตรวจสอบความถูกต้องในส่วน PROCEDURE DIVISION

ปัญหาในส่วนนี้ สำหรับ IDENTIFICATION DIVISION และ ENVIRONMENT DIVISION อาจจะไม่มีปัญหาที่ยุ่งยากเนื่องจากการตรวจสอบนั้นส่วนใหญ่ใน 2 ส่วนดังกล่าวเป็นไปในลักษณะจากบนลงล่าง (Top-Down Parsing) ทั้งสิ้น กล่าวคือ เมื่อพบคำสงวน (Reserved Word) คำใดคำหนึ่งแล้วก็สามารถคาดหวังได้ว่าสิ่งต่อไปที่จะพบคืออะไรบ้าง ซึ่งมีไม่มากกรณีนัก

ส่วน DATA DIVISION นั้นจำเป็นที่จะต้องเก็บตัวแปรต่าง ๆ ซึ่งประกอบด้วย

1. ชื่อตัวแปร
2. ลักษณะของตัวแปรนั้น ได้แก่
   - 2.1 ประเภทของตัวแปรซึ่งแบ่งเป็น
     - 2.1.1 ตัวแปรจำนวน (Numeric Dataname)
     - 2.1.2 ตัวแปรอักขระ (Alphabetic Dataname)
     - 2.1.3 ตัวแปรที่มีค่าได้ทั้งอักขระและตัวเลข (Alphanumeric Dataname)
     - 2.1.4 ตัวแปรแถวลำดับ (Array)
     - 2.1.5 ตัวแปรที่เก็บค่าตกแต่ง (Editted)
   - 2.2 ความยาวของตัวแปร
   - 2.3 เครื่องหมายบอกว่าเป็นตัวแปรกลุ่ม (Group Item) หรืออาจจะเป็นลักษณะของตัวแปรเดี่ยว (Elementary Item)

ค่าเริ่มต้นของตัวแปรที่ถูกกำหนดด้วยวลี VALUE ไม่จำเป็นต้องเก็บเข้าตารางสัญญลักษณ์ หรือตารางค่าคงที่ (Literal Table) เนื่องจากวัตถุประสงค์ของระบบเพียงแต่ตรวจสอบความถูกต้องของไวยากรณ์ ดังนั้นจึงทำหน้าที่เพียงตรวจสอบว่า ค่าเริ่มต้นที่กำหนดนั้นสอดคล้องกับประเภทของตัวแปรนั้นหรือไม่ ซึ่งถ้าไม่สอดคล้องก็จะเป็นความผิดพลาดที่จะต้องแจ้งให้ผู้ใช้ทราบ

ใน PROCEDURE DIVISION นั้น จะต้องตรวจความถูกต้องของคำสั่งที่เขียนขึ้น แยกตามชนิดของคำสั่ง โดยคำสงวนในภาษาโคบอลนั้นแยกเป็นประเภทต่าง ๆ ได้แก่

1. <u>คำสั่ง</u> (Verb หรือ Command) คำสงวนประเภทนี้ถือเป็นคำสำคัญที่แสดงความหมายของประโยคคำสั่ง เมื่อถูกแปลเป็นภาษาเครื่องก็จะเป็นรหัสคำสั่งที่แท้จริง ที่กำหนดขั้นตอนการทำงาน (Operation Code) เช่น ADD, MOVE, SUBTRACT

2. <u>คำประกอบ</u> คำสงวนแบบนี้กำหนดขึ้นเพียงเพื่อทำให้ประโยคคำสั่งต่าง ๆ ในภาษาโคบอลมีรูปแบบที่ดูคล้ายกับลักษณะของประโยคภาษาอังกฤษมากขึ้นเท่านั้น โดยทั่วไปแล้ว คำประกอบนี้ไม่ว่าจะเขียนหรือไม่เขียนก็จะไม่ทำให้ความหมายของคำสั่งนั้นเปลี่ยนแปลงไป ตัวอย่างเช่น

```
2.1 ...IF VARIABLE-1 IS EQUAL TO VARIABLE-2
          THEN DISPLAY 'TWO VAIRABLES ARE EQUAL.'.

2.2 ...IF VARIABLE-1 EQUAL VARIABLE-2
          DISPLAY 'TWO VARIABLES ARE EQUAL.'.
```

ประโยคคำสั่งทั้งสอง ในประโยคที่ 1 มีคำประกอบได้แก่ IS, TO, THEN ซึ่งประโยคคำสั่งทั้งสองประโยคนั้นมีความหมายเหมือนกันทุกประการ

3. <u>คำแสดงความสัมพันธ์ของโปรแกรมกับสภาพแวดล้อม</u> คำประเภทนี้เพื่อเป็นการให้ข่าวสารแก่ระบบปฏิบัติการ (Operating System) ถึงความจำเป็นที่จะเกี่ยวข้องกับอุปกรณ์รับส่งข้อมูล โดยที่คำประเภทนี้มักเขียนแสดงขึ้นภายใน ENVIRONMENT DIVISION ได้แก่ วลี SELECT หรือย่อหน้า SPECIAL-NAMES

4. คำแสดงข่าวสารแก่ตัวแปลโปรแกรม (Compiler Directive) มักแสดงอยู่ใน DATA DIVISION เพื่อบอกตัวแปลโปรแกรมให้มีการจัดเตรียมเนื้อที่ในหน่วยความจำขนาดเท่าไร และจะมีการใช้เนื้อที่ในหน่วยความจำ เป็นลักษณะอย่างไร ตัวอย่างของคำประเภทนี้คือ วลี PICTURE, VALUE, REDEFINES, OCCURS นอกจากคำประเภทนี้จะเกี่ยวข้องกับการเตรียมเนื้อที่ในหน่วยความจำแล้วยังเป็นข่าวสารแก่ตัวแปรโปรแกรมในการจัดสร้างตารางสัญญลักษณ์ด้วย

สำหรับ PROCEDURE DIVISION นั้น การตรวจสอบไวยากรณ์ของคำสงวนประเภทคำสั่งและคำประกอบโดยทั่วไปก็ยังคงเป็นไปตามแบบบนลงล่าง กล่าวคือสามารถที่จะคาดหวังว่าเมื่อพบโทเกนคำสั่งนี้แล้วต่อไปจะพบโทเกนอะไรที่ตามมาบ้าง

ปัญหาบางอย่างสำหรับการวิเคราะห์ความถูกต้องของไวยากรณ์ คือ

1. คำสั่งควบคุมลำดับการทำงาน ได้แก่ GO และ PERFORM สามารถอ้างชื่อกระบวนการ (Procedure-name) ได้ก่อนที่จะพบชื่อกระบวนการนั้น แสดงตัวอย่างคือ

```
        ADD SALES-AMOUNT TO ACCUMULATED-SALES-AMOUNT.
        PERFORM CHECK-FOR-WARNING-ROUTINE.
        IF  WARNING-FLAG-IS-ON
            THEN...
             :
CHECK-FOR-WARNING-ROUTINE.
             :
```

ดังนั้นในขณะที่ตรวจคำสั่งประเภทนี้ อาจจะยังไม่รู้ว่า ชื่อกระบวนการนั้นมีอยู่จริงหรือไม่

2. คำสั่งแสดงการเปรียบเทียบ (IF Statement) อาจจะถูกเขียนขึ้นในลักษณะตาข่าย (Nested IF) ซึ่งต้องมีการติดตามคำสั่งนี้ว่า คำสั่งแสดงทางเลือกอีกทาง (ELSE) มีคู่กันอย่างเหมาะสมหรือไม่

3. คำสั่งคำนวณ COMPUTE ใช้รูปแบบนิพจน์ทางคณิตศาสตร์ ดังนั้นต้องมีวิธีการตรวจสอบนิพจน์ทางคณิตศาสตร์ด้วย

ข้อมูลที่จำเป็นในการสร้างเฟสที่ 2

ในเฟสการวิเคราะห์ความถูกต้องของไวยากรณ์ภาษา มีข้อมูลต่าง ๆ ที่ใช้ ได้แก่

1. ข้อมูลโทเกนที่สร้างมาจากเฟสการวิเคราะห์เลกซิเคิล
2. บัฟเฟอร์ของแฟ้มข้อมูล CBLSK4.TKN ที่เก็บข้อมูลโทเกน โดยกำหนดใช้ในส่วนเนื้อที่ DMA ของระบบ
3. ตัวเก็บรหัสความผิดพลาด (ERRCD) เป็นตัวแปรตัวเดียวกันกับที่ใช้ในเฟสการวิเคราะห์เลกซิเคิล
4. บัฟเฟอร์ของแฟ้มข้อมูลที่ใช้เก็บรหัสความผิดพลาดจากการวิเคราะห์ในเฟสที่ 2 โดยจะเก็บรหัสความผิดพลาดดังกล่าวลงในแฟ้มข้อมูลชื่อ CBLSK5.ERR
5. ตัวแปรใช้เก็บชื่อของส่วนที่โทเกนนั้นปรากฏในโปรแกรมดิบ (AREA) เพื่อใช้ตรวจสอบว่าโทเกนนั้นอยู่ในเนื้อที่ที่ถูกต้องหรือไม่โดยมีกฎเกณฑ์ดังนี้

   5.1 ต้องขึ้นต้นในส่วนเนื้อที่ A คือ

   - ชื่อ ดิวิชัน (DIVISION)
   - ชื่อ เซกชัน (SECTION)
   - ชื่อ ย่อหน้า (Paragraph)
   - ตัวชี้ลำดับ FD
   - ตัวชี้ลำดับ SD
   - ตัวชี้ลำดับ RD
   - ตัวชี้ลำดับ CD

- ตัวชี้ลำดับ 01
- ตัวชี้ลำดับ 77

5.2 ต้องขึ้นต้นในส่วนเนื้อที่ B คือ

- เนื้อหาในส่วน IDENTIFICATION DIVISION
- เนื้อหาในส่วน ENVIRONMENT DIVISION
- คำหรือวลีในลำดับ FD, SD, RD, CD
- เนื้อหาอื่นใดใน DATA DIVISION
- ประโยคคำสั่งใน PROCEDURE DIVISION
- การต่อของคำสั่งแต่ละบรรทัด

5.3 ขึ้นต้นในส่วน A หรือ B ก็ได้

- เลขแสดงลำดับ 02-49
- เลขแสดงลำดับ 66
- เลขแสดงลำดับ 88

6. ตัวแปรเก็บรหัสของคำสงวนที่ได้จากการค้นหาในตารางคำสงวน (Reserved Word Table) (NOTE)
7. ตารางคำสงวนเก็บคำสงวนทุกคำที่ใช้ในระบบ IBM-VS-COBOL พร้อมทั้งรหัสคำสงวนนั้น (RSVTAB)
8. เนื้อที่ใช้เก็บตารางสัญญลักษณ์ (SYMTAB) ซึ่งใช้เนื้อที่ในหน่วยความจำทั้งหมดที่เหลือจากการเก็บรหัสเครื่อง ของโปรแกรมตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล
9. ตัวแปรที่ใช้เก็บบันทึกเพื่อบอกว่าโทเกนที่อ่านเข้ามานั้นเป็นประเภทของโทเกนคำสั่งหรือไม่ (VERB)
10. ตารางเก็บเฉพาะคำสงวนประเภทคำสั่ง (VERB00)
11. ตัวแปรเก็บหมายเลขลำดับบรรทัด (COUNT1)
12. ตัวแปรเก็บตัวชี้ (Pointer) ของตัวแปรย่อยที่ชี้ไปยังที่อยู่ (Address) ของตัวแปรกลุ่มที่คุมตัวแปรย่อยนั้นในตารางสัญญลักษณ์ (KTRACK)
13. ตารางเก็บตัวชี้ (Pointer) ของตัวแปรแต่ละลำดับ (01, 02-49) ในกรณี

ที่มีตัวแปรถูกสร้างขึ้นด้วยเลขลำดับหลายลำดับ (TRKTAB)

14. บัฟเฟอร์ของระบบเพื่อใช้ในการดูล่วงหน้า (Look ahead) สำหรับโทเกนที่ต้องการ (BUFFER)

รูปแบบและโครงสร้างของข้อมูลที่ใช้ในเฟสที่ 2

ข้อมูลโทเกนที่สร้างมาจากเฟสการวิเคราะห์เลกซิเคิล

ข้อมูลโทเกน คือข้อมูลในแฟ้ม CBLSK4.TKN ที่สร้างมาจากเฟสที่ 1 ดังนั้นรูปแบบก็คือรูปแบบเดียวกันกับที่ได้นำเสนอในหัวข้อลักษณะแฟ้มข้อมูลโทเกนของการสร้างเฟสที่ 1

บัฟเฟอร์ของแฟ้มข้อมูลโทเกน

เป็นเนื้อที่ในหน่วยความจำ แอดเดรส 080H หรือ DMA ของระบบซีพีเอ็มนั่นเอง เป็นเนื้อที่ในหน่วยความจำที่มีความยาว 128 ไบต์ ใช้สำหรับส่วนทำงานย่อยชื่อ FRNOTE ซึ่งทำหน้าที่รับรู้ทีละโทเกน

บัฟเฟอร์ของแฟ้มข้อมูลเก็บรหัสความผิดพลาด

ใช้เนื้อที่บัฟเฟอร์เดียวกับ CBLSK3.ERR ในเฟสที่ 1 ที่ชื่อ BUFFER แต่ในเฟสที่ 2 นี้จัดเตรียมให้เนื้อที่ส่วนดังกล่าวนั้นกลายเป็นบัฟเฟอร์ของแฟ้มข้อมูลที่เก็บรหัสความผิดพลาดของเฟสที่ 2 นี้ ชื่อ CBLSK5.ERR โดยลักษณะของ CBLSK5.ERR จะมีลักษณะเหมือนกับลักษณะของ CBLSK3.ERR ทุกประการ

ตารางคำสงวนภาษาโคบอล

ได้ถูกสร้างไว้เป็นเนื้อที่ภายนอกโปรแกรม (External) ในลักษณะของโปรแกรม

ภาษาแอสเซมบลี Z-80 ที่ให้เฉพาะส่วนของข้อมูล (Data segment) โดยไม่มีส่วนของรหัสการทำงาน (Code segment) ในรูปของรหัสที่สามารถเคลื่อนย้ายได้ (Relocatable code) ทั้งนี้เพื่อความสะดวกในการเชื่อมต่อ (Link) โปรแกรมในเฟสที่ 2 เข้ากับตารางคำสงวนภาษาโคบอล

ตารางคำสงวนภาษาโคบอลถูกแบ่งออกเป็น 15 ส่วนตามความยาวของคำสงวน ซึ่งมีความยาวตั้งแต่ 1-15 ตัวอักษร ทั้งนี้เพื่อความรวดเร็วในการค้นหาตารางคำสงวน โดยใช้การค้นหาแบบดรรชนี (Index) มีความยาวของโทเกนเป็นดรรชนี

ทุกส่วนของตารางคำสงวนภาษาโคบอลมีการจัดเนื้อหาเหมือนกัน กล่าวคือ

1. เนื้อหาของคำสงวน ความยาวขึ้นอยู่กับความยาวของคำสงวน
2. รหัสประจำคำสงวนคำนั้นมีความยาว 2 ไบต์เสมอ

A	T	00H	01H	B	Y	00H	02H	...

ในการค้นหาคำในตารางคำสงวนนั้น ในทางปฏิบัติ ยังต้องอาศัยตารางประกอบอีกตารางหนึ่ง คือตารางเก็บจำนวนคำสงวนแต่ละส่วนความยาว (NUMBER) โดยสมาชิกที่ 1 ของตารางจำนวน แสดงถึงคำสงวนที่มีความยาว 1 ไบต์, สมาชิกที่ 2 ของตารางจำนวน แสดงถึงคำสงวนที่มีความยาว 2 ไบต์ ฯลฯ

07H	19H	26H	32H	30H	3BH	...

ตารางเก็บจำนวนคำสงวนดังกล่าวมี 15 สมาชิก แต่ละสมาชิกใช้เนื้อที่ 1 ไบต์

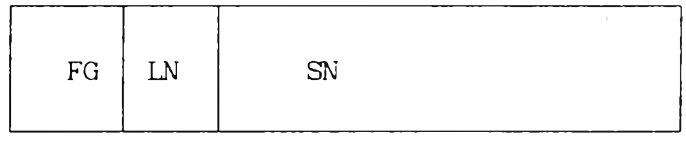
โดยเก็บจำนวนของคำสงวนตามความยาวในรูปแบบเลขฐานสอง ดังนั้นความยาวของตารางใช้เนื้อที่ทั้งสิ้น 15 ไบต์

ตารางสัญญลักษณ์

เป็นตารางที่ถูกสร้างขึ้นในขณะที่ตัวตรวจสอบทำงานถึงเฟสที่ 2 ในส่วนของ DATA DIVISION โดยมีวัตถุประสงค์เพื่อเก็บตัวแปร (Data name) และสัญญลักษณ์อื่น ๆ เช่น ชื่อเงื่อนไข (Condition-name) เป็นต้น

รูปแบบและวิธีการเก็บตารางสัญญลักษณ์สามารถแสดงได้ 2 ประเภทดังนี้

ประเภทที่ 1 ได้จาก IDENTIFICATION DIVISION และ ENVIRONMENT DIVISION



FG - เป็นตัวบอก (Flag) ว่าสัญญลักษณ์ที่จะตามมานั้นเป็นสัญญลักษณ์ประเภทใด โดย FG ใช้เนื้อที่ 1 ไบต์ ซึ่งมีรายละเอียดดังนี้

- 00H หมายถึง สัญญลักษณ์นั้นเป็นชื่อโปรแกรม (PROGRAM-ID)
- 01H หมายถึง สัญญลักษณ์นั้นเป็นชื่อพิเศษ (SPECIAL-NAMES)
- 02H หมายถึง สัญญลักษณ์นั้นเป็นชื่อแฟ้มข้อมูลนำเข้า
- 03H หมายถึง สัญญลักษณ์นั้นเป็นชื่อแฟ้มข้อมูลนำออก

LN - เป็นความยาวของชื่อสัญญลักษณ์ ที่จะตามมา จะเก็บในรูปของเลขฐานสอง ใช้เนื้อที่ 1 ไบต์

SN - เป็นชื่อของสัญญลักษณ์ (Symbol Name) ที่ผู้เขียนโปรแกรมเป็นผู้กำหนด

ประเภทที่ 2 ได้จาก DATA DIVISION

FG	AD	AN	LD	LN	SN

FG - เป็นตัวบอกว่าสัญญลักษณ์ที่จะตามมานั้นเป็นสัญญลักษณ์ประเภทใด ใช้เนื้อที่ 1 ไบต์ ซึ่งมีรายละเอียดดังนี้

- 04H หมายถึง สัญญลักษณ์นั้น เป็นชื่อเงื่อนไข ที่ถูกสร้างขึ้น โดยการใช้เลขลำดับ 88
- 05H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรตัวเลข (Numeric)
- 06H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรอักขระ (Alphabetic)
- 07H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรผสม (Alphanumeric)
- 08H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 1 ลำดับ โดยเป็นประเภทตัวเลข (1-Dimensional numeric array)
- 09H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 1 ลำดับ โดยเป็นประเภทตัวอักษร (1-Dimensional alphabetic array)
- 0AH หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 1 ลำดับ โดยเป็นประเภทผสม (1-Dimensional alphanumeric array)
- 0BH หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 2 ลำดับ โดยเป็นประเภทตัวเลข (2-Dimensional numeric array)
- 0CH หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 2 ลำดับ โดยเป็นประเภทตัวอักษร (2-Dimensional alphabetic array)
- 0DH หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 2 ลำดับ โดยเป็นประเภทผสม (2-Dimensional alphanumeric array)
- 0EH หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 3 ลำดับ โดยเป็นประเภทตัวเลข (3-Dimensional numeric array)

- 0FH หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 3 ลำดับ โดยเป็นประเภทตัวอักษร (3-Dimensional alphabetic array)
- 10H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรแถวลำดับ 3 ลำดับ โดยเป็นประเภทผสม (3-Dimensional alphanumeric array)
- 11H หมายถึง สัญญลักษณ์นั้นเป็นชื่อตัวแปรประเภทมีลักษณะการตกแต่ง (Editted Dataname)
- 12H หมายถึง สัญญลักษณ์นั้นเป็นชื่อระเบียนข้อมูลจากแฟ้มข้อมูลนำเข้า (Input record)
- 13H หมายถึง สัญญลักษณ์นั้นเป็นชื่อระเบียนข้อมูลจากแฟ้มข้อมูลนำออก (Output record)

AD - ตัวชี้ไปยังแอดเดรสของตัวแปรกลุ่ม ที่อยู่เหนือขึ้นไปจากตัวแปรตัวนั้น ใช้เนื้อที่ 2 ไบต์ โดยอาจจะมีค่าเป็น 00H 00H ในกรณีที่เป็นตัวแปรอิสระ (เลขลำดับ 77 แสดงว่าเป็น Independent dataname) หรืออาจจะเป็นชื่อเงื่อนไข (สร้างโดยเลขแสดงลำดับ 88)

AN - เป็นตัวชี้ไปยังแอดเดรสของตัวแปร ที่มีความยาวเดียวกัน ที่อยู่ถัดไปตามตรรก (Logical Adjacency) ถ้ามีค่าเป็น 00H แสดงว่าตัวนั้นเป็นสัญญลักษณ์ตัวสุดท้ายของความยาวนั้น ใช้เนื้อที่ในการเก็บ 2 ไบต์

LD - เก็บความยาวของตัวแปร (ค่าของตัวแปรมีความยาวได้เท่าไร) ใช้เนื้อที่เก็บ 2 ไบต์ ในรูปของเลขฐานสอง

LN - เก็บความยาวของชื่อตัวแปรหรือสัญญลักษณ์ที่จะตามมาใช้ความยาว 1 ไบต์

SN - เก็บชื่อของสัญญลักษณ์ ความยาวขึ้นอยู่กับ LN

<u>ตารางเก็บคำสงวนประเภทคำสั่ง</u>

เป็นตารางที่ใช้เก็บบันทึกรหัสคำสงวน เฉพาะประเภทของคำสงวนที่เป็นคำสั่งใน PROCEDURE DIVISION โดยเก็บเฉพาะรหัสคำสงวนเท่านั้น แต่ละสมาชิกใช้เนื้อที่ในหน่วยความจำ มีความยาว 2 ไบต์ โดยแบ่งเป็น 2 ส่วน

ส่วนที่ 1 คือพวกคำสงวนที่มีรหัสคำสงวนในไบต์ที่ 1 เป็นค่า 00H

ส่วนที่ 2 คือพวกคำสงวนที่มีรหัสคำสงวนในไบต์ที่ 1 เป็นค่า 01H

00H		00H		...	01H		01H	.	...

ตารางเก็บตัวชี้ของตัวแปรแต่ละลำดับ (TRKTAB)

เป็นตารางที่สร้างขึ้นในขณะตรวจสอบ DATA DIVISION โดยมีลักษณะเป็นตารางพลวัตร (Dynamic Table) กล่าวคือ ข้อมูลในตารางเปลี่ยนแปลงตลอดเวลา ลักษณะของข้อมูลในตารางจะเก็บแอดเดรสของตัวแปรกลุ่ม (Group item) แต่ละลำดับ (Level)

- สมาชิกที่ 1 ของตารางเก็บแอดเดรสของตัวแปรกลุ่มลำดับ 01 ในขณะนั้น (Current Level 01 dataname Address)
- สมาชิกที่ 2 ของตารางเก็บแอดเดรสของตัวแปรกลุ่มลำดับ 02 ในขณะนั้น (Current Level 02 dataname Address)

:

- สมาชิกที่ 49 ของตารางเก็บแอดเดรสของตัวแปรกลุ่มลำดับ 49 ในขณะนั้น (Current Level 49 dataname Address)

บัฟเฟอร์ของระบบเพื่อใช้ดูโทเคนล่วงหน้า (BUFFER)

เนื่องจากในระบบการตรวจสอบไวยากรณ์ ส่วนของ PROCEDURE DIVISION มีความจำเป็นที่จะต้องดูโทเคนล่วงหน้าว่าเป็นอะไร ในทางปฏิบัติการดูโทเคนล่วงหน้านั้นก็เหมือนกับการรับรู้โทเคนนั่นเอง แต่ต้องไม่เลื่อนตัวชี้ (Pointer) ไปยังโทเคนต่อไป

(Next token) ดังนั้นอาจเกิดปัญหา ถ้าเป็นโทเกนสุดท้ายของดีเอ็มเอ เพราะจะต้องอ่านข้อมูลโทเกนระเบียนใหม่เข้ามาแทนที่ในดีเอ็มเอ ทั้งระเบียน ซึ่งจะทำให้ตัวชี้ชี้โทเกนตัวต่อไปผิดที่หมด

ด้วยปัญหาดังกล่าว จึงมีความจำเป็นต้องสร้างบัฟเฟอร์ ของระบบ อีกตัวหนึ่งเพื่อพักข้อมูลโทเกนถัดไปจากการอ่านข้อมูลโทเกนเข้ามาใหม่ทั้งระเบียนเข้ามาไว้ในบัฟเฟอร์ดังกล่าว (BUFFER) เพื่อที่จะไม่ทำลายข้อมูลเก่าในดีเอ็มเอ ซึ่งเก็บข้อมูลโทเกนที่ยังจำเป็นต้องใช้อยู่

รูปแบบของ BUFFER จึงเหมือนกับดีเอ็มเอ ใช้เนื้อที่ในหน่วยความจำ ความยาว 128 ไบต์

ตัวเก็บค่าและติดตามที่จำเป็นอื่น ๆ ในเฟสที่ 2

ในเฟสที่ 2 นอกจากเนื้อที่ที่ถูกจัดสรรให้เป็นบัฟเฟอร์และเป็นตารางต่าง ๆ แล้ว ยังต้องจัดสรรเนื้อที่ในหน่วยความจำเพื่อเก็บค่าต่าง ๆ ดังนี้

1. ตัวเก็บรหัสความผิดพลาด (ERRCD) ใช้เก็บรหัสความผิดที่ตรวจพบเป็นตัวเดียวกันกับที่ใช้ในเฟสที่ 1 มีความยาว 2 ไบต์
2. ตัวเก็บแสดงส่วนที่อยู่ของโทเกนที่รับรู้ได้ (AREA) ซึ่ง AREA นี้จะมีค่าเพียง A หรือ B เท่านั้น ใช้เนื้อที่ 1 ไบต์
3. ตัวแปรเก็บค่าที่แสดงว่าโทเกนที่รับรู้ได้นั้นเป็นคำสั่งหรือไม่ (VERB) โดยถ้ามีค่าเป็น T หมายถึง โทเกนนั้นเป็นคำสงวนประเภทคำสั่งใน PROCEDURE DIVISION ใช้เนื้อที่ 1 ไบต์
4. ตัวแปรเก็บหมายเลขบรรทัด (COUNT1) เป็นตัวแปรที่ใช้เพื่อวัตถุประสงค์ในการเก็บหมายเลขบรรทัดอ้างอิงเช่นเดียวกับเฟสที่ 1 แต่ว่าเป็นตัวแปรคนละตัวกันที่มีลักษณะเหมือนกันทุกประการ (เก็บตัวเลขแสดงได้ในรหัส ASCII และใช้ความยาว 4 ไบต์) ทั้งนี้ในเฟสที่ 2 COUNT1 จะเก็บค่าที่รับรู้

ได้จากส่วนทำงานย่อย FRNOTE

5. ตัวแปรเก็บค่าตัวชี้แอดเดรส (KTRACK) เพื่อใช้เก็บแอดเดรส ของตัวแปรกลุ่มเป็นการชั่วคราวก่อนจะเก็บลงใน TRKTAB

<u>วิธีการและอัลกอริธึมที่ใช้ในเฟสที่ 2</u>

หน้าที่หลักของเฟสที่ 2 คือการตรวจสอบลำดับของโทเกน ที่อ่านเข้ามา จากแฟ้มข้อมูล CBLSK4.TKN โดยแบ่งเฟสนี้เป็น 5 ส่วน ได้แก่

1. <u>ส่วนทำงานเริ่มต้น</u> (Initialization)

ในส่วนนี้จะทำหน้าที่โดยทั่ว ๆ ไปก่อนที่จะเริ่มการตรวจสอบความถูกต้องของโปรแกรมคิบจริง ๆ หน้าที่ดังกล่าวประกอบด้วย

1.1 เปิดแฟ้มข้อมูล CBLSK4.TKN

1.2 เตรียมสร้างแฟ้มข้อมูล CBLSK5.ERR เพื่อเก็บรหัสความผิดพลาดที่ตรวจพบได้ในเฟสนี้

1.3 อ่านข้อมูลจาก CBLSK4.TKN เข้ามาระเบียนแรก

1.4 เรียกส่วนทำงานย่อย FRNOTE เพื่อรับรู้รหัสของโทเกนตัวแรกว่าเป็น "IDENTIFICATION" หรือไม่ แล้วจะส่งการทำงานให้ส่วนต่อไป

2. <u>ส่วนตรวจสอบ IDENTIFICATION DIVISION</u> (IDDIV)

ในส่วนนี้ จะทำการตรวจสอบโดยหลักการบนลงล่าง (Top down parsing) กล่าวคือ เมื่อพบรหัสโทเกนที่แทนคำว่า "IDENTIFICATION" แล้วโทเกนต่อไปต้องเป็นรหัสแทนคำว่า "DIVISION" และ "." ตามลำดับ มิฉะนั้นจะมีการเรียกส่วนทำงานย่อย ERROR เพื่อให้บันทึกความผิดพลาดลงในแฟ้ม CBLSK5.ERR โดยรหัสความผิดพลาดจะถูกส่งผ่านโดยตัวแปร ERRCD

นอกจากนั้นส่วนนี้ยังจะทำหน้าที่เก็บชื่อโปรแกรม (PROGRAM-ID) ลงเก็บในตารางสัญญลักษณ์พร้อมกับรหัส 00H ด้วย เพื่อบอกว่าสัญญลักษณ์ตัวนี้คือชื่อโปรแกรม

ในส่วนนี้จะทำกระบวนการดังกล่าวเรื่อยไปจนกว่าจะพบกับรหัสโทเกน 01H 058H ซึ่งแทนคำว่า "ENVIRONMENT" ก็จะส่งการทำงานต่อให้ส่วนต่อไป

3. ส่วนตรวจสอบ ENVIRONMENT DIVISION (ENVIRO)

ส่วนนี้จะตรวจสอบความถูกต้องของโปรแกรมดิบ ENVIRONMENT DIVISION โดยใช้หลักการเช่นเดียวกับส่วนก่อนทุกประการ และจะเก็บสัญญลักษณ์ต่าง ๆ (ถ้ามี) เข้าสู่ตารางสัญญลักษณ์ ได้แก่

3.1 ชื่อพิเศษ โดยใช้รหัส 01H นำหน้า

3.2 ชื่อแฟ้มข้อมูลนำเข้า ใช้รหัส 02H นำหน้า

3.3 ชื่อแฟ้มข้อมูลนำออก ใช้รหัส 03H นำหน้า

ส่วนตรวจสอบ ENVIRONMENT DIVISION นี้จะทำงานตรวจสอบไปจนกว่าจะพบกับโทเกน "DATA" อันเป็นจุดเริ่มต้นของ DATA DIVISION แล้วจึงจะผ่านการทำงานไปยังส่วนถัดไป

4. ส่วนตรวจสอบ DATA DIVISION และการเก็บค่าตารางสัญญลักษณ์

ส่วนนี้แบ่งหน้าที่หลักเป็น 2 อย่าง ได้แก่

4.1 การตรวจสอบความถูกต้อง ของการเขียนโปรแกรม ในส่วนของ DATA DIVISION (DATADV) หน้าที่ดังกล่าวนี้ก็ยังใช้หลักการตรวจสอบจากบนลงล่างเหมือนกับส่วนก่อน ๆ

4.2 การเก็บค่าสัญญลักษณ์ต่าง ๆ เข้าตารางสัญญลักษณ์ (KPTSYM) ในส่วน

KPTSYM มีความยุ่งยากในการสร้างโปรแกรมมากพอสมควร กล่าวคือ จะต้องติดตามตัวชี้แอดเดรสของตัวแปรกลุ่ม ต้องติดตามตัวชี้แอดเดรสของตัวแปรที่มีความยาวของชื่อตัวแปรเท่ากัน เพื่อที่จะเก็บลงในตารางสัญญลักษณ์

นอกจากนั้น ยังต้องมีการตรวจสอบวลีต่าง ๆ ได้แก่

- PICTURE
- USAGE
- SIGN
- OCCURS
- SYNCHRONIZED
- JUSTIFIED
- BLANK WHEN ZERO
- VALUE
- REDEFINES
- RENAMES

ซึ่งบางวลี มีผลต่อรหัสของประเภทของตัวแปร ที่จะต้องจัดเก็บเข้าในตารางสัญญลักษณ์ เช่น วลี OCCURS จะทำให้ตัวแปรที่เก็บอาจจะมีรหัสประเภทเป็น 08H, 09H, 0AH, 0BH, 0CH, 0DH, 0EH, 0FH, 10H อย่างใดอย่างหนึ่ง

วลี PICTURE ทำให้การเก็บรหัสต้องคำนึงถึง 3 สิ่งคือ

4.2.1 ตัวแปรนั้นเป็นตัวแปรย่อย

4.2.2 ความยาวของค่าของตัวแปรนั้น โดยจะเรียกส่วนทำงานย่อย CONVRT แปลงค่าตัวเลขแสดงความยาวในวลี PICTURE ที่เขียนด้วยเลขฐานสิบ ให้อยู่ในรูปของเลขฐานสองเพื่อความ

สะดวกในการเก็บค่าในตารางสัญญลักษณ์

4.2.3 ทำให้รู้ถึงประเภทของตัวแปรว่าเป็นตัวแปรตัวเลข ตัวแปรอักขระ หรือตัวแปรผสม

การทำงานของส่วนตรวจสอบ DATA DIVISION นี้ จะทำไปจนกว่าจะพบกับโทเกน "PROCEDURE" จึงจะส่งผ่านการทำงานให้ส่วนถัดไป

5. ส่วนการตรวจสอบ PROCEDURE DIVISION (PROCDE)

ในส่วนที่ 5 นี้ มีสิ่งที่จะต้องพิจารณาคือ

5.1 ในคำสั่งทั่วไปของ PROCEDURE DIVISION ก็ยังสามารถใช้หลักการตรวจสอบแบบบนลงล่างได้ เช่น คำสั่ง

MOVE [{ CORR / CORRESPONDING }] identifier - 1 TO identifier-2.

เมื่อพบ "MOVE" ต่อไปอาจจะพบ "CORR" หรือไม่ก็ได้ ("CORR" กับ "CORRESPONDING" จะมีรหัสโทเกนแทนด้วยรหัสเดียวกัน คือ 00H 45H) จากนั้นต้องพบกับ identifier-1 และ ฯลฯ

5.2 คำสั่งเกี่ยวกับการควบคุมลำดับการทำงาน ได้แก่คำสั่ง GO (TO) และคำสั่ง PERFORM อาจจะอ้างชื่อกระบวนการที่ยังทำงานไปไม่ถึง ดังนั้นเมื่อพบคำสั่ง GO TO หรือ PERFORM ต้องเก็บชื่อกระบวนการนั้นไปค้นหาในตาราง PRCTAB ที่เก็บชื่อกระบวนการของส่วนของโปรแกรมใน

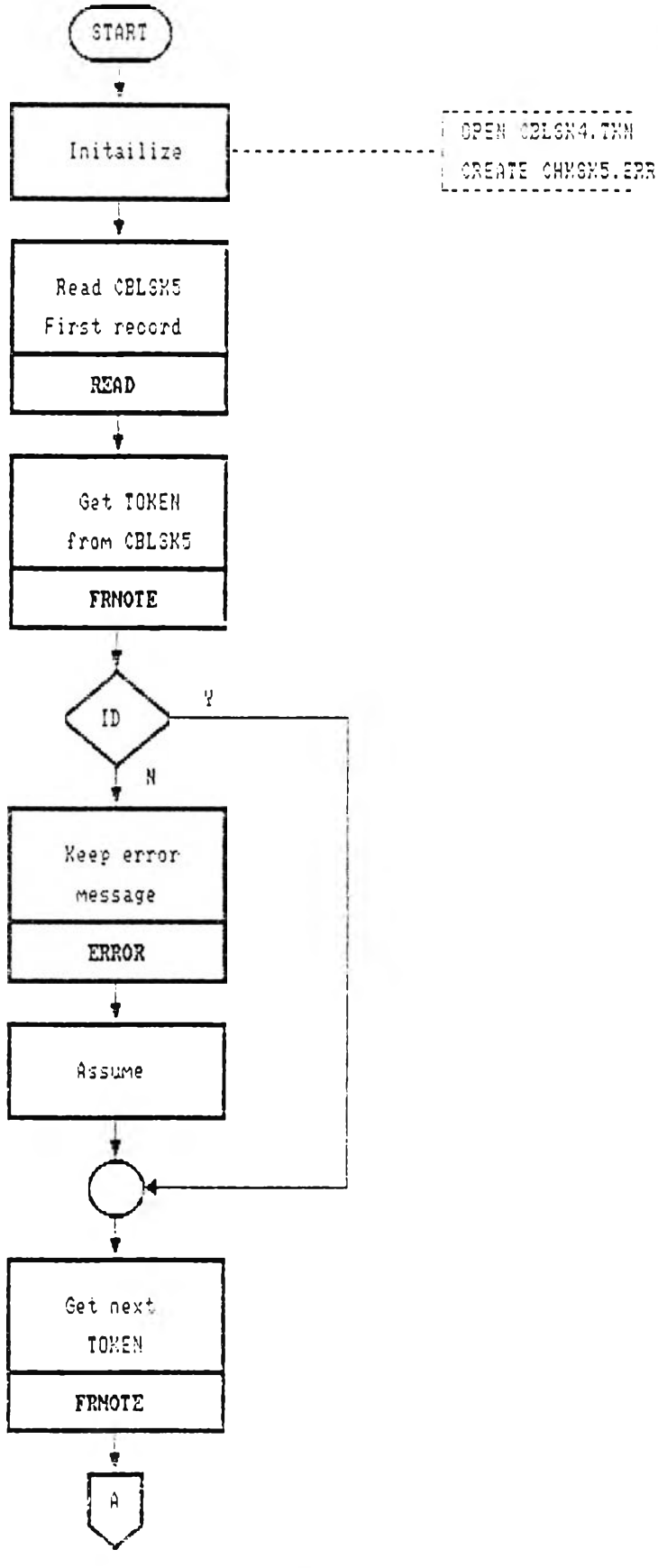
PROCEDURE DIVISION ถ้าหากพบก็ถือว่าคำสั่งนั้นถูกต้อง แต่ถ้าหากไม่พบ ก็จะต้องเก็บชื่อกระบวนการหลังคำสั่งดังกล่าวพร้อมทั้งหมายเลขลำดับเข้าสู่ตาราง PNDTAB หลังจากได้ตรวจจนหมดทุกโทเกน (หมายถึงหมดข้อมูลในแฟ้มข้อมูล CBLSK4.TKN) แล้วจึงนำค่าในตาราง PNDTAB ทีละตัวมาตรวจสอบกับค่าในตาราง PRCTAB ถ้าจับคู่หรือตรวจหาไม่พบ ก็จะสร้างรหัสความผิดพลาดขึ้น

5.3 คำสั่งแสดงการคำนวณนิพจน์ COMPUTE ใช้หลักการตรวจสอบนิพจน์โดยทั่วไปคือเปลี่ยนเป็นรูปแบบที่แสดงค่านิพจน์ (Polish Notation) ในที่นี้ได้เลือกการใช้แบบสัญญลักษณ์เครื่องหมายอยู่หลัง (Postfix) โดยทำการตรวจสอบนิพจน์ทางคณิตศาสตร์ ซึ่งอาศัยกฎรูปแบบ "Well-formed expression" ที่กล่าวว่า "A Polish suffix (prefix) formula is well-formed if and only if the rank of the formula is "one" and the rank of any proper head of a polish formula is greater than (less than) or equal to "one""

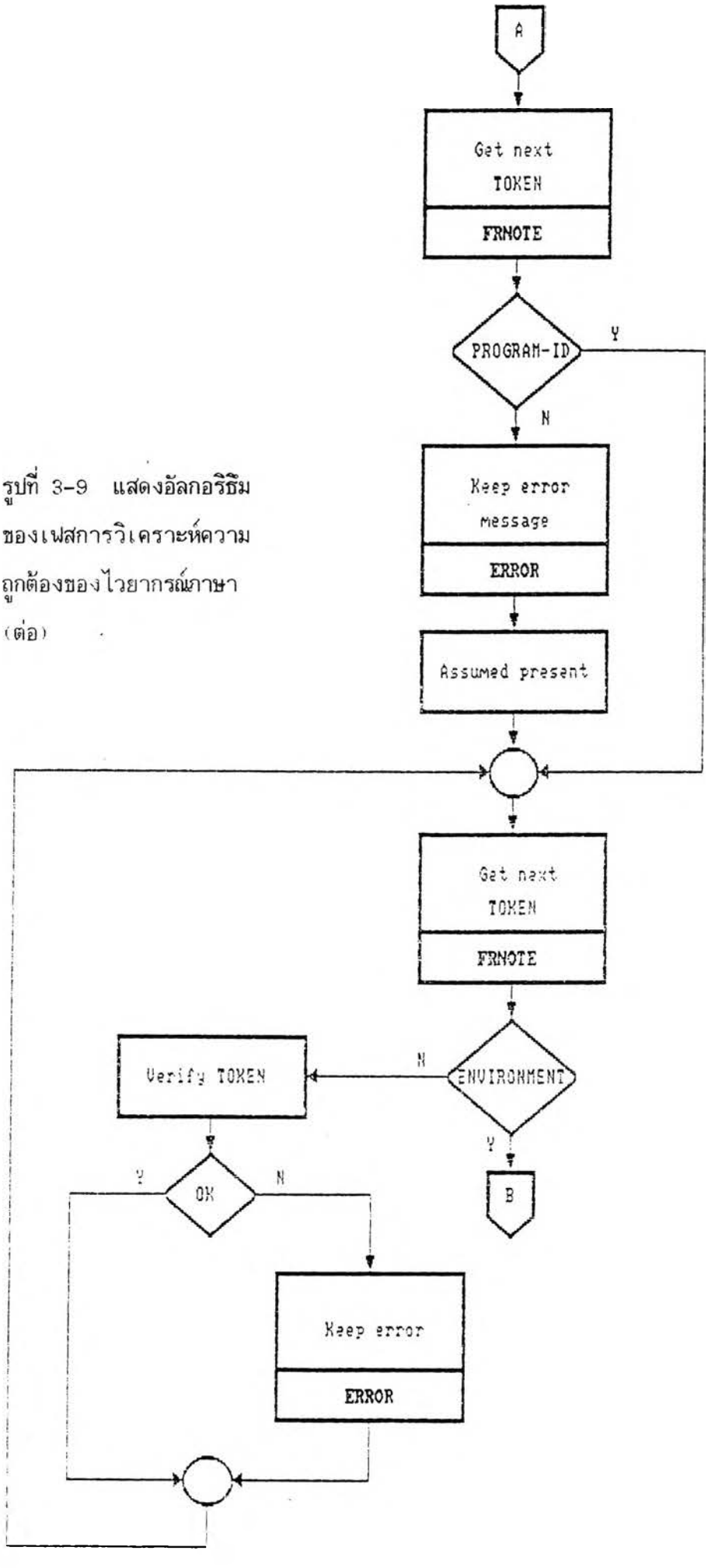
5.4 คำสั่งตรวจสอบเงื่อนไข IF...THEN...ELSE ในกรณีที่มีการทำตาข่าย IF (Nested IF) ก็จะใช้ตัวนับ IFCNT เป็นตัวติดตามนับจำนวนการใช้คำสั่ง IF กล่าวคือถ้าพบการใช้คำสั่ง IF ก็จะบวกสะสมค่า 1 เข้าไปยัง IFCNT เมื่อพบ ELSE 1 ตัว ก็จะลบค่า IFCNT ออกด้วย 1 จนถึงเมื่อจบประโยค ก็จะตรวจดูว่า IFCNT มีค่าเป็นบวกหรือไม่ ถ้ายังเป็นบวกก็ยังถูกต้อง แต่ถ้าน้อยกว่า 0 แสดงว่ามี ELSE มากกว่า IF

6. <u>ส่วนทำงานสุดท้าย</u> (Finalization)

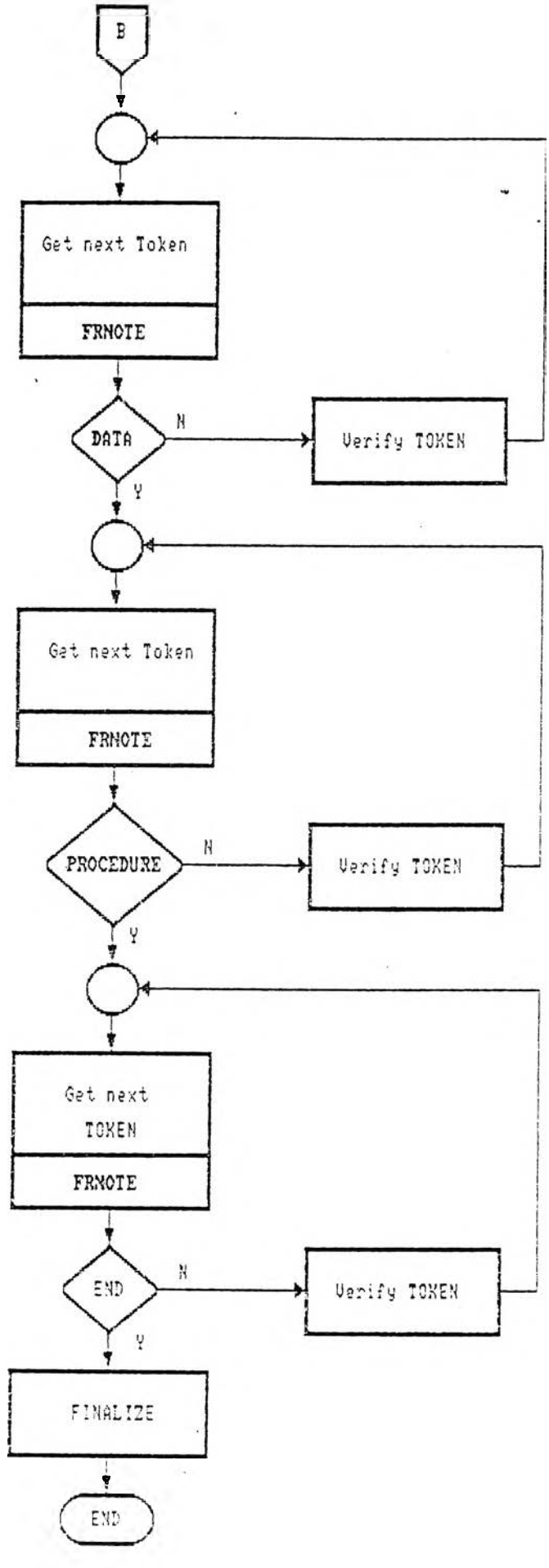
มีชื่อว่า END ทำหน้าที่



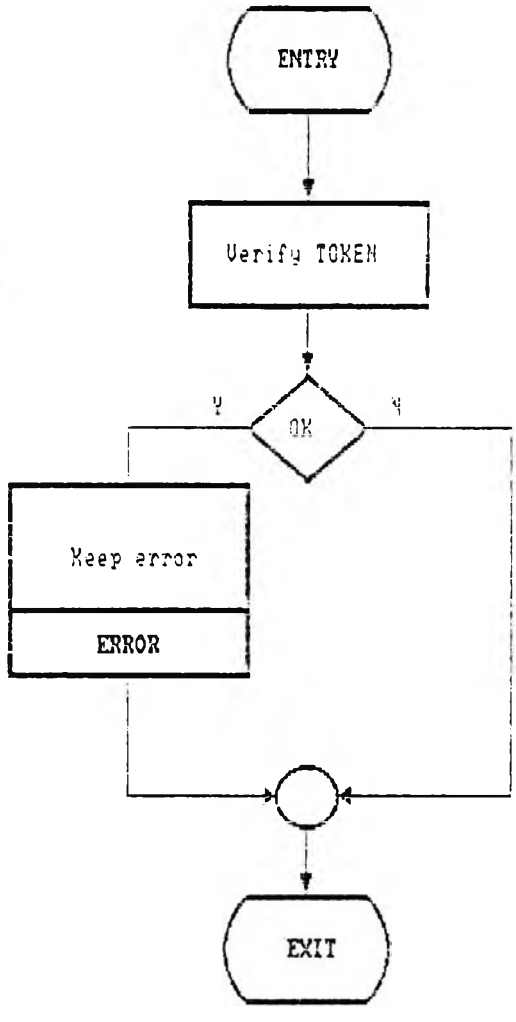
รูปที่ 3-9 แสดงอัลกอริธึมของเฟสการวิเคราะห์ความถูกต้องของไวยากรณ์ภาษา



รูปที่ 3-9 แสดงอัลกอริธึม
ของเฟสการวิเคราะห์ความ
ถูกต้องของไวยากรณ์ภาษา
(ต่อ)



รูปที่ 3-9 แสดงอัลกอริธึมของเฟสการวิเคราะห์ความถูกต้องของไวยากรณ์ภาษา (ต่อ)



รูปที่ 3-10 แสดงอัลกอริธึมในการตรวจสอบโทเกนของแต่ละดิวิชัน

6.1 แสดงว่าจบเฟสที่ 2 ขึ้นทางหน้าจอ

6.2 ปิดแฟ้มข้อมูล CBLSK4.TKN และ CBLSK5.ERR

จะเห็นได้ว่าการทำงานในเฟสที่ 2 นี้เป็นการวิเคราะห์ความถูกต้องทางไวยากรณ์ของภาษา (Syntactical Analysis) เท่านั้น โดยไม่สนใจความหมายของภาษา (Semantic) ว่าการเขียนจะให้ความหมายถูกต้องหรือไม่เพียงใด

การแบ่งส่วนย่อยในเฟสที่ 2

การแบ่งส่วนย่อยในเฟสนี้ สามารถพิจารณาได้ว่า ในเฟสนี้จะประกอบด้วยส่วนย่อยต่าง ๆ ได้แก่

1. IDDIV ทำหน้าที่ตรวจสอบ IDENTIFICATION DIVISION
2. ENVIRO ทำหน้าที่ตรวจสอบ ENVIRONMENT DIVISION
3. DATADV ทำหน้าที่ตรวจสอบ DATA DIVISION
4. PROCDE ทำหน้าที่ตรวจสอบ PROCEDURE DIVISION
5. FRNOTE ทำหน้าที่รับรู้โทเคน แล้วส่งค่ารหัสโทเคนมาในตัวแปร NOTE ส่งค่า A หรือ B มาในตัวแปร AREA และส่งค่า T หรือ N มาในตัวแปร VERB และความยาวโทเคนมาในตัวแปร LENGTH
6. AHEAD ทำหน้าที่ดูโทเคนล่วงหน้า แล้วส่งค่ากลับมาเหมือนกับส่วนทำงานย่อย FRNOTE ทุกประการ แต่ไม่เลื่อนตัวชี้ ไปยังโทเคนตัวถัดไป
7. DTNAME ทำหน้าที่ตรวจสอบว่าโทเคนที่พบนั้น (รีจิสเตอร์ HL จะเป็นตัวชี้ว่าโทเคนนั้นมีเนื้อหาอยู่ที่ไหน) ถูกต้องตามกฎการตั้งชื่อของตัวแปร ของภาษาโคบอลหรือไม่
8. DATNAM เป็นส่วนการทำงานย่อยทำหน้าที่ ในการตรวจหาสัญญลักษณ์ที่ต้องการในตารางสัญญลักษณ์
9. LITERL เป็นส่วนทำงานย่อยตรวจสอบว่าสายวลีที่ต้องการตรวจสอบนั้นเป็น

ค่าคงที่ประเภทใด หรือผิดกฎเกณฑ์ ไม่สามารถเป็นค่าคงที่ได้

10. READ ทำหน้าที่อ่านข้อมูลจาก CBLSK4.TKN
11. ERROR เป็นส่วนทำงานย่อยเก็บรหัสความผิดพลาด เหมือนกับเฟสที่ 1 ทุกประการ
12. KPTSYM ทำหน้าที่เก็บค่าสัญญลักษณ์เข้าสู่ตารางสัญญลักษณ์
13. PICTUR ตรวจสอบว่าสายวลีที่พบนั้นเป็นสายวลีที่ถูกต้องสำหรับวลี PICTURE หรือไม่ ถ้าถูกต้องเป็นประเภทใด
14. CONVRT เปลี่ยนค่าเลขฐานสิบให้เป็นค่าในเลขฐานสอง

เฟสที่ 3 การสร้างข่าวสารความผิดพลาด

นิยามของปัญหา

เฟสการสร้างข่าวสารความผิดพลาด เป็นเฟสสุดท้ายของการสร้างตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล ซึ่งใน 2 เฟสแรกนั้นจะสร้างข้อมูลนำเข้าให้เฟสนี้ ได้แก่

1. โปรแกรมต้นภาษาโคบอลที่ผู้ใช้เขียนขึ้น
2. รหัสความผิดพลาด มี 2 แฟ้มข้อมูล
   2.1 จากเฟสการวิเคราะห์เลกซิเคิล CBLSK3.ERR
   2.1 จากเฟสการวิเคราะห์ความถูกต้องของไวยากรณ์ CBLSK5.ERR

ในเฟสนี้ จะนำข้อมูลดังกล่าวนั้น มาสร้างเป็นข่าวสารความผิดพลาด เพื่อผู้ใช้งานสามารถเรียกดูได้จากแฟ้ม CBLSK9.PRN หรืออาจจะพิมพ์แสดงผลออกทางเครื่องพิมพ์ก็ได้

ข้อมูลที่จำเป็นในการสร้างเฟสที่ 3

การสร้างข่าวสารความผิดพลาดนี้ มีข้อมูลที่จำเป็นในการสร้าง ได้แก่

1. ข้อมูลโปรแกรมต้นภาษาโคบอล จากแฟ้ม CBLSK1.PRN

2. ข้อมูลรหัสความผิดพลาด จากแฟ้ม CBLSK3.ERR
3. ข้อมูลรหัสความผิดพลาด จากแฟ้ม CLKSK5.ERR
4. ผลลัพธ์ข่าวสารความผิดพลาดพร้อมกับโปรแกรมดิบ ในแฟ้ม CBLSK9.PRN
5. ตารางข่าวสารความผิดพลาด (MESSGE)
6. เนื้อที่เก็บรหัสความผิด (DATA)
7. ตัวเก็บแอดเดรสของสมาชิกสุดท้ายของเนื้อที่เก็บรหัสความผิด (KLAST)
8. เนื้อที่ชั่วคราว ใช้เก็บข่าวสารความผิด ก่อนถ่ายเทลงบัฟเฟอร์ เพื่อบันทึกลงแผ่นจานแม่เหล็ก (MSG1)

รูปแบบและโครงสร้างข้อมูลที่ใช้ในเฟสที่ 3

ข้อมูลโปรแกรมดิบภาษาโคบอล (CBLSK1.PRN)

เป็นโปรแกรมภาษาโคบอลที่ผู้ใช้งานเขียนขึ้น และได้มีการคัดลอกในเฟสที่ 1 ลงมาเก็บไว้ในแฟ้มข้อมูล CBLSK1.PRN ในลักษณะแฟ้มข้อมูลของระบบบีซีพีเอ็ม ซึ่งถูกจัดเก็บในแผ่นจานแม่เหล็กดังแสดงในรูป

เนื้อหาข้อมูล 80 ไบต์	เนื้อที่เหลือใช้ 48 ไบต์	1 เซกเตอร์

ดังนั้นในการอ่าน 1 ครั้ง ซึ่งจะต้องอ่านทีละ 1 เซกเตอร์จะได้ข้อมูลที่ใช้เนื้อง 80 ไบต์ โดยเก็บในลักษณะรหัส ASCII

ข้อมูลรหัสความผิดพลาดจากเฟสที่ 1 (CBLSK3.ERR)

คือรหัสความผิดพลาดที่ตรวจพบได้ในเฟสที่ 1 บันทึกในแฟ้มข้อมูล CBLSK3.ERR มี

รูปแบบดังได้แสดงมาแล้วในเฟสที่ 1

ข้อมูลรหัสความผิดพลาดเฟสที่ 2 (CBLSK5.ERR)

คือรหัสความผิดพลาดที่ตรวจพบได้ในเฟสที่ 2 บันทึกในแฟ้มข้อมูล CBLSK5.ERR มีรูปแบบดังได้แสดงมาแล้วในเฟสที่ 2

ผลลัพธ์ข่าวสารความผิดพลาดพร้อมโปรแกรมคีบ (CBLSK9.PRN)

เป็นส่วนนำออกของเฟสที่ 3 และเป็นวัตถุประสงค์หลักของตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล โดยจะมีลักษณะเป็นแฟ้มข้อมูล ภายใต้ระบบซีนีเอ็ม

1 เซกเตอร์	เนื้อหาข้อมูล 1	ODH	OAH	เนื้อหาข้อมูล 2	ODH	OAH ..

กล่าวคือใน 1 เซกเตอร์ บนแผ่นจานแม่เหล็ก อาจจะเก็บได้มากกว่า 1 ระเบียน โดยในส่วนของเนื้อหาข้อมูลนั้น มีความยาวไม่คงที่ แล้วแต่ความมากน้อยของเนื้อหาที่จะบันทึก ในตอนท้ายระเบียนจะมีการเติมค่า ODH และ OAH 2 ไบต์เพื่อเป็นการบอกจุดสิ้นสุดของข้อมูลแต่ละระเบียน

แฟ้มข้อมูลดังกล่าวนี้ ใช้ชื่อว่า CBLSK9.PRN

ตารางข่าวสารความผิดพลาด (MESSGE)

เป็นตารางที่ประกอบไปด้วยเนื้อหา 2 ส่วน ได้แก่

1. รหัสความผิดพลาด ใช้เนื้อที่ 2 ไบต์ เป็นรหัสความผิดพลาดเหมือนกับที่อยู่ใน CBLSK3.ERR และ CBLSK5.ERR

2. ส่วนข่าวสารความผิดพลาด แบ่งเป็น 2 ส่วนย่อย ได้แก่
   2.1 ส่วนบอกความยาว ใช้เนื้อที่ 1 ไบต์เพื่อบอกว่าข่าวสารที่จะตามมานั้นมีความยาวเท่าไร เก็บบันทึกในลักษณะเลขฐานสอง
   2.2 ส่วนข่าวสาร เป็นส่วนที่เก็บข้อความ ที่แสดงความผิดพลาดชนิดต่าง ๆ ความยาวจะถูกกำหนดมาจากส่วนบอกความยาว ที่เก็บบันทึกในลักษณะตัวอักษร โดยรหัส ASCII

00H	02H	0FH	S	e	q	u	e	n	c	e		e	r	r	o	r	.

เนื้อที่เก็บรหัสความผิดพลาด 'DATA'

ใช้เนื้อที่ในหน่วยความจำหลัก ส่วนที่อยู่ต่อจากโปรแกรมเพื่อเก็บข้อมูลที่อ่านจาก CBLSK3.ERR และ CBLSK5.ERR แล้วจึงนำมาเรียงลำดับตามลำดับหมายเลขบรรทัด เพื่อนำไปใช้สร้างข่าวสารความผิดพลาดต่อไป

เนื้อที่ชั่วคราวเก็บข่าวสารความผิดพลาด 'MSG1'

เป็นเนื้อที่เตรียมไว้ใช้ชั่วคราว ก่อนถ่ายเทไปยังบัฟเฟอร์ เพื่อจะใช้เตรียมบันทึกลงในแผ่นจานแม่เหล็ก ประกอบด้วย 2 ส่วนคือ

1. ส่วนเก็บหมายเลขบรรทัด ใช้เนื้อที่ 6 ไบต์
2. ส่วนเก็บข้อความลักษณะความผิด ใช้เนื้อที่ 80 ไบต์ และเติม 0DH 0AH อีก 2 ไบต์

ตัวเก็บแอดเดรสของสมาชิกสุดท้ายของเนื้อที่เก็บรหัสความผิด (KLAST)

เป็นตัวแปรที่ใช้เก็บค่าแอดเดรสของสมาชิกตัวสุดท้ายไบต์สุดท้าย ที่ใช้งานของเนื้อที่เก็บรหัสความผิด ใช้เนื้อที่ 2 ไบต์ เพื่อใช้ประโยชน์ 2 ประการ

1. รู้จุดสิ้นสุดของกระบวนการจัดเรียงลำดับ
2. รู้จุดสิ้นสุดของการสร้างข่าวสารความผิดพลาด

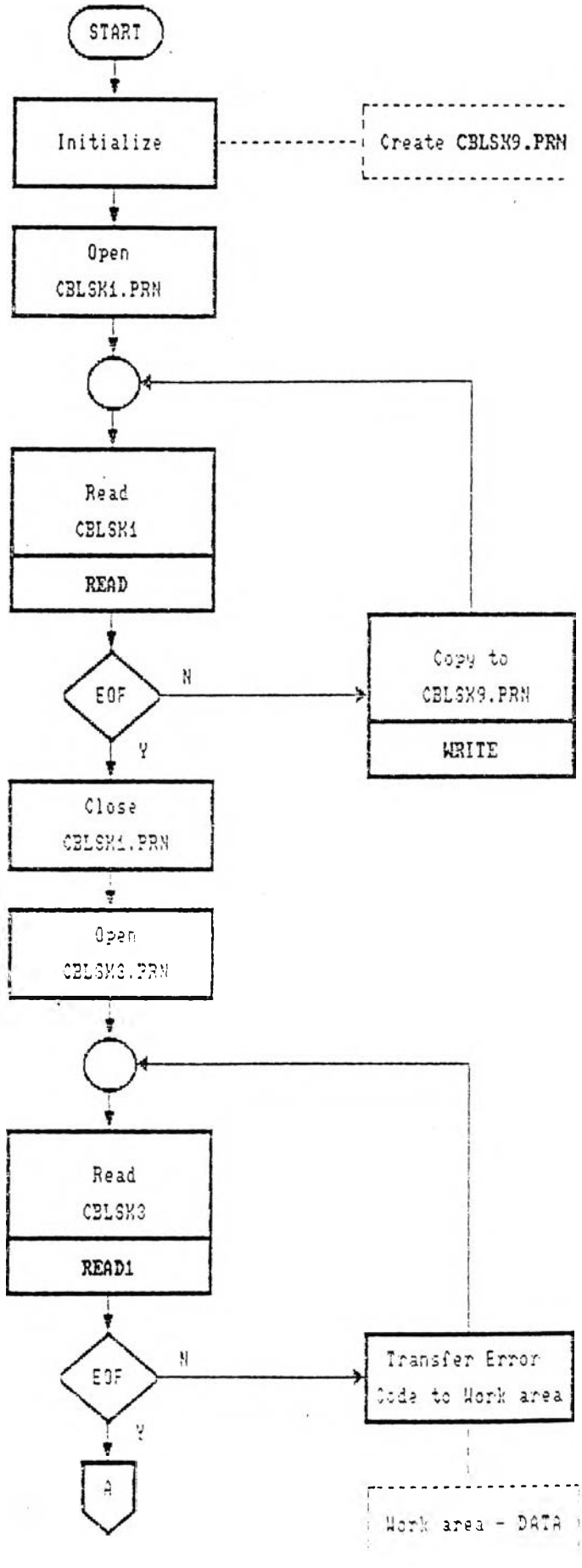
วิธีการและอัลกอริธึมที่ใช้ในเฟสที่ 3

ในเฟสที่ 3 นี้ เป็นการสร้างข่าวสารความผิดพลาดจากรหัสความผิดพลาดโดยอาจจะนำเสนอวิธีการโดยแบ่งเป็นขั้นตอนได้ดังนี้

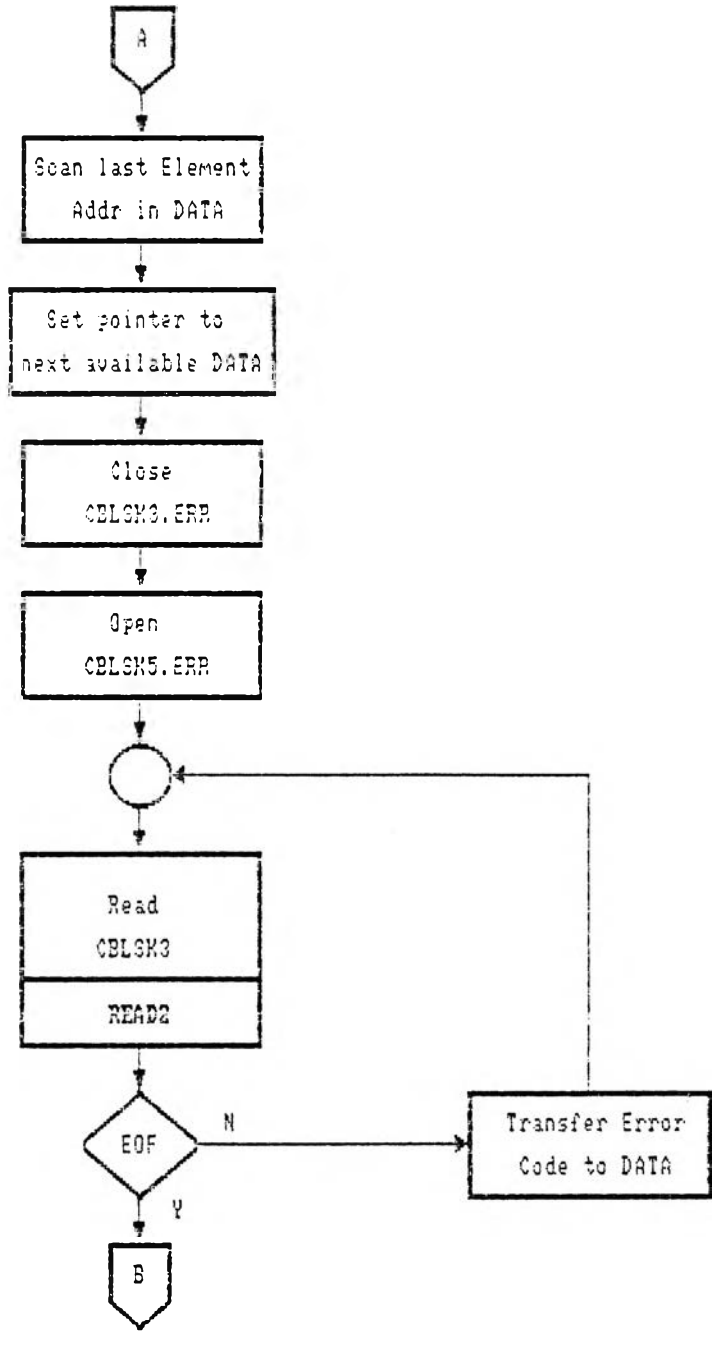
1. ขั้นตอนการลอกโปรแกรมดิบจากแฟ้ม CBLSK1.PRN มาสู่แฟ้ม CBLSK9.PRN ขั้นตอนนี้ทำเพียงแต่อ่านข้อมูลจากแฟ้ม CBLSK1.PRN มาทีละ 1 เซกเตอร์ ซึ่งจะได้มา 1 ระเบียนหรือ 1 บรรทัดของโปรแกรมดิบ แล้วนำมาผ่านกระบวนการลดขนาดของระเบียน (Compress) โดยหาตัวอักษรตัวแรกที่ไม่ใช่ช่องว่าง จากคอลัมน์ที่ 80 ของแต่ละบรรทัดขึ้นมาสู่คอลัมน์ต้น เพื่อลดขนาดของระเบียนลง จากนั้นจึงเติมอักษร 0AH และ 0DH เพื่อบอกจุดจบของระเบียน แล้วจึงถ่ายทอดลงในบัฟเฟอร์ เพื่อเตรียมบันทึกลงในแผ่นจานแม่เหล็ก

2. ขั้นตอนการอ่านรหัสความผิดพลาด เป็นการอ่านข้อมูลรหัสความผิดพลาดจากแฟ้มข้อมูล CBLSK3.ERR และ CBLSK5.ERR แล้วนำมาไว้ในหน่วยความจำหลัก โดยจะผ่านกระบวนการจัดการลบเครื่องหมายของจุดจบของแฟ้มข้อมูล (ที่ได้เติม * ไว้จนเต็ม) ออกจากพื้นที่ DATA

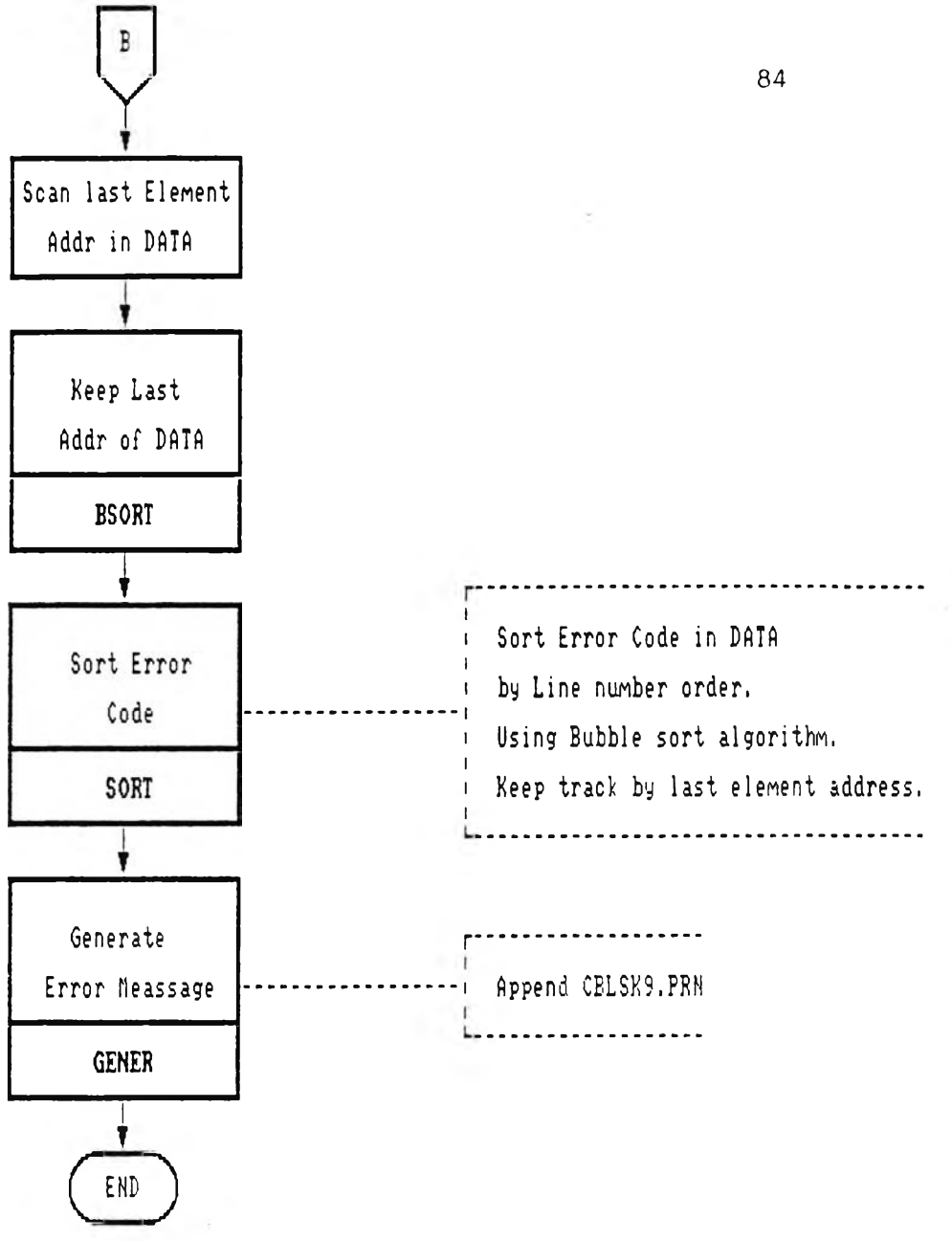
3. การจัดเรียงลำดับตามหมายเลขบรรทัด หลังจากอ่านรหัสความผิดพลาดมาต่อไว้ใน DATA แล้ว จะทำการเรียงลำดับรหัสความผิดพลาดแต่ละสมาชิก ตามลำดับหมายเลขบรรทัด โดยใช้การเปลี่ยนที่ (Interchange Sort) อัลกอริธึมที่ใช้เป็นการเรียงลำดับแบบ



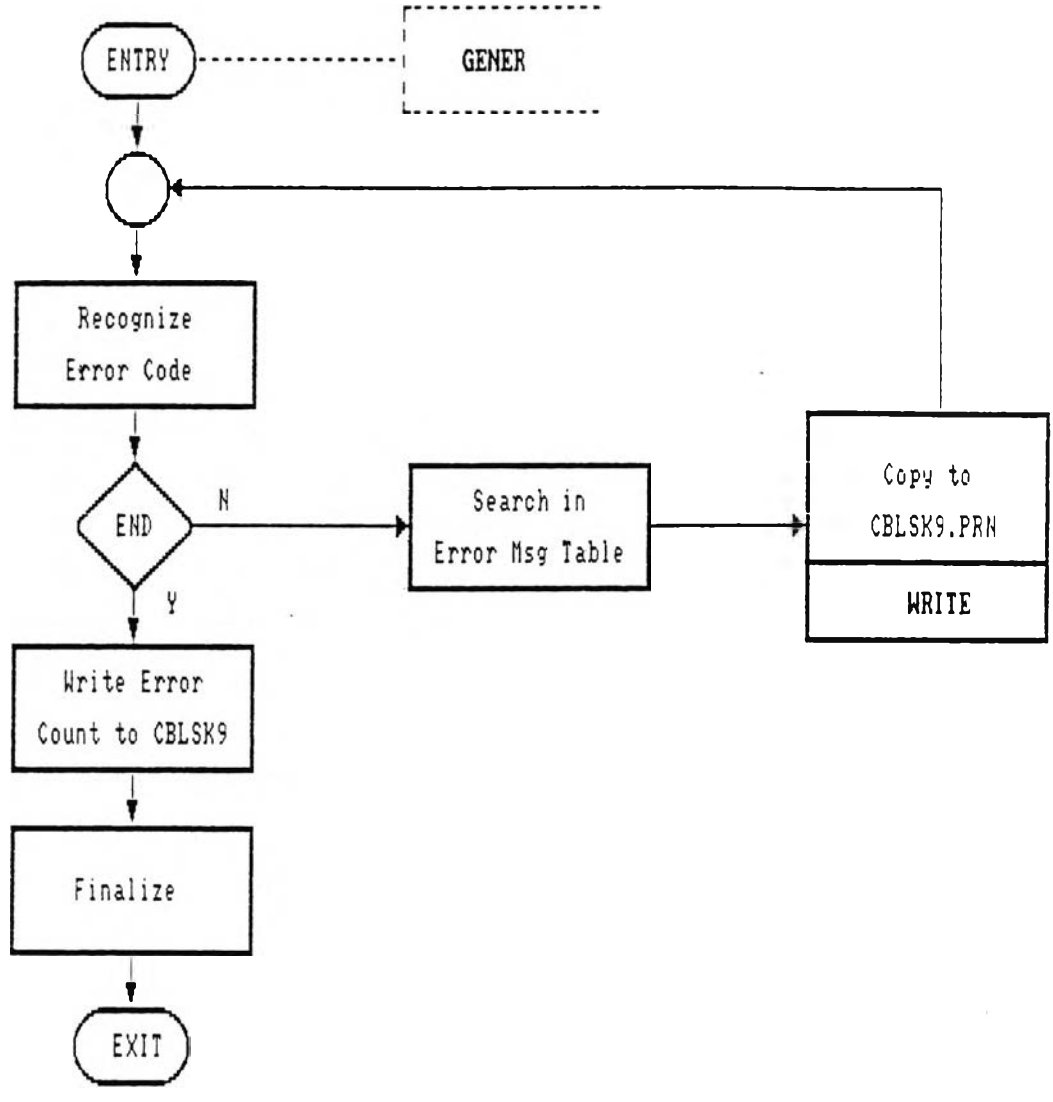
รูปที่ 3-11
แสดงอัลกอริธึม
ในการสร้าง
ข่าวสารความผิดพลาด



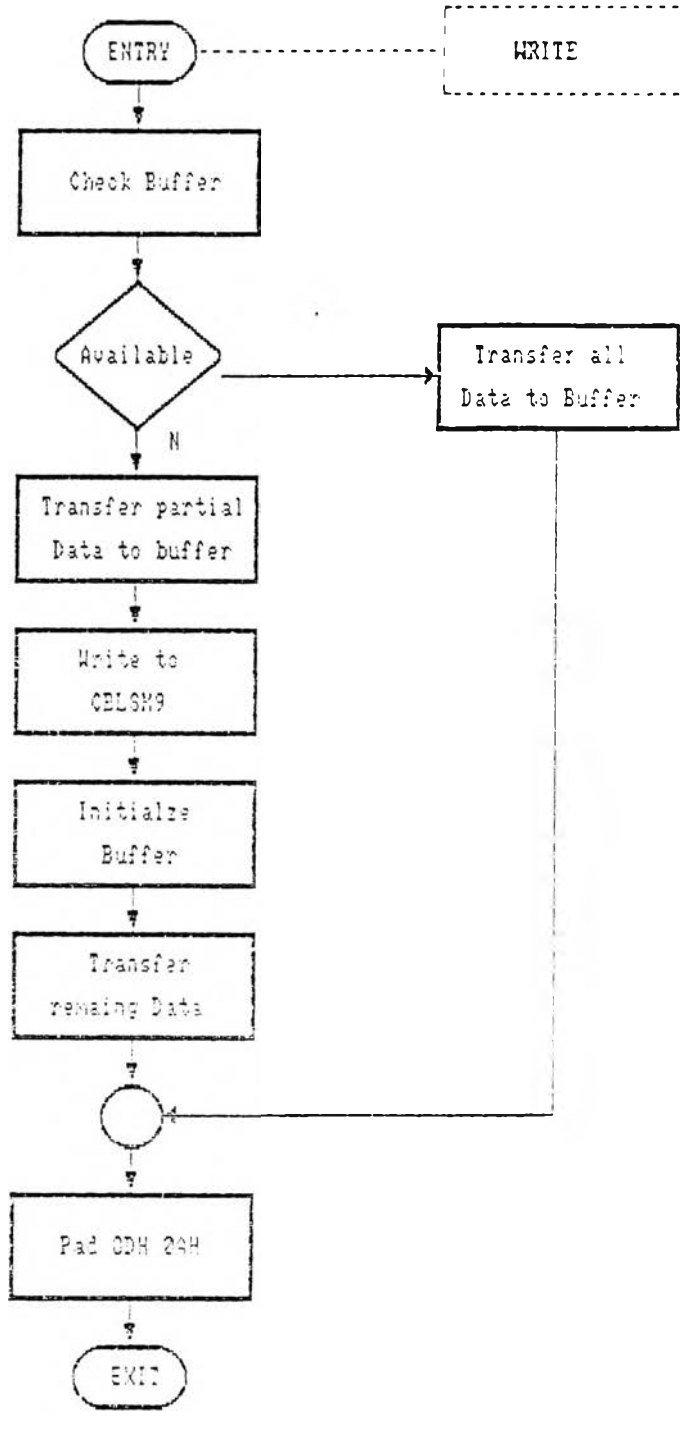
รูปที่ 3-11 แสดงอัลกอริธึมในการสร้างข่าวสารความผิดพลาด (ต่อ)



รูปที่ 3-11 แสดงอัลกอริธึมในการสร้างข่าวสารความผิดพลาด (ต่อ)



รูปที่ 3-12 แสดงอัลกอริธึมของโปรแกรม GENER



รูปที่ 3-13 แสดงอัลกอริธึมของโปรแกรม WRITE

ลอยตัว (Bubble Sort.) ซึ่งคำศัพท์ KLAST เป็นตัวกำหนดขอบเขตการเรียงลำดับ

4. การค้นหาตารางข่าวสารความผิดพลาด เมื่อเรียงลำดับรหัสความผิดพลาดแล้ว จะนำรหัสมาค้นหา ในตารางข่าวสารความผิดพลาด แล้วจึงนำข่าวสารนั้นมาจัดเตรียมลงใน บัฟเฟอร์ เพื่อบันทึกลงแผ่นจานแม่เหล็กต่อไป

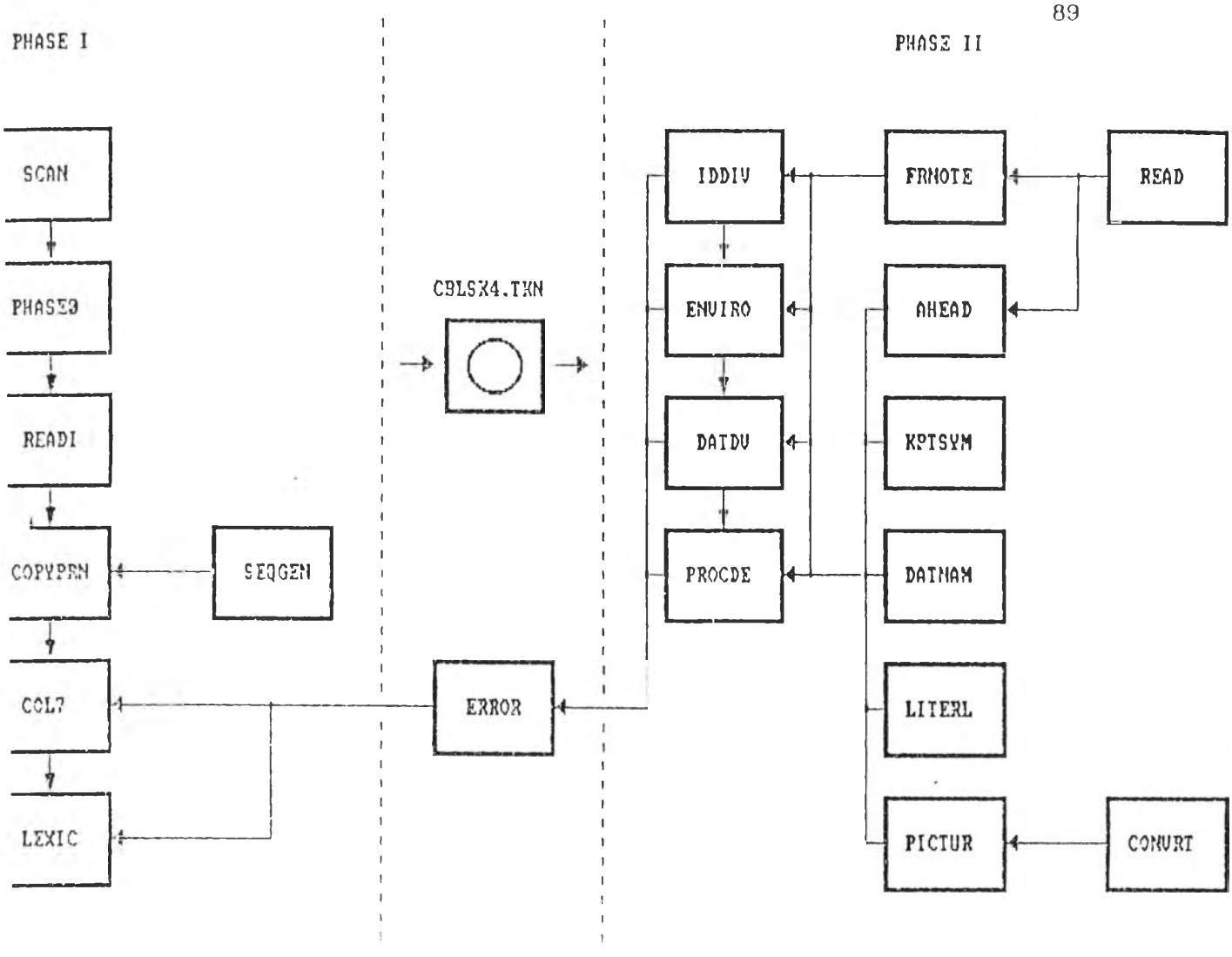
การแบ่งส่วนย่อยในเฟสที่ 3

ในเฟสที่ 3 นี้ ถูกแบ่งออกเป็นโมดูลย่อยสำหรับประมวลผลต่าง ๆ ดังนี้

1. READ ทำหน้าที่อ่านข้อมูลจาก CBLSK1.PRN เข้าสู่ระบบ
2. WRITE ทำหน้าที่จัดการบัฟเฟอร์ของแฟ้มข้อมูล CBLSK9.PRN โดยจะนำเอา ข้อมูลในบัฟเฟอร์ เก็บบันทึกลงบนแผ่นจานแม่เหล็ก แล้วทำการให้ค่าเริ่มต้น แก่บัฟเฟอร์ใหม่อีกครั้ง
3. READ1 ทำหน้าที่อ่านข้อมูลจาก CBLSK3.ERR แล้วย้ายเข้าสู่พื้นที่ DATA
4. BREADX ทำหน้าที่อ่านข้อมูล CBLSK3.ERR และหาแอดเดรสในพื้นที่ DATA ต่อไปที่สามารถจะบันทึกข้อมูลจากแฟ้ม CBLSK5.ERR (Next. available area)
5. READ2 ทำหน้าที่อ่านข้อมูลจาก CBLSK5.ERR แล้วย้ายเข้าสู่พื้นที่ DATA
6. BSORT ทำหน้าที่หาแอดเดรสสุดท้าย ของสมาชิกรหัสความผิดพลาดตัวสุดท้าย ในพื้นที่ DATA
7. SORT จัดเรียงลำดับรหัสความผิดพลาด ตามหมายเลขบรรทัดคำสั่งที่ผิด
8. GENER นำรหัสความผิดพลาด ที่ได้จากการเรียงลำดับแล้วนั้น มาค้นหาใน ตารางข่าวสารความผิดพลาด แล้วนำข่าวสารความผิดพลาดที่ค้นพบ มาบันทึก ในแผ่นจานแม่เหล็ก แฟ้มข้อมูล CBLSK9.PRN
9. SCHEAD ทำหน้าที่สร้างระเบียนหัวรายงานของแต่ละหน้ารายงาน

ทั้งหมดที่นำเสนอเป็นการออกแบบสร้างตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอล โดยแบ่งการออกแบบเป็น 3 เฟส ซึ่งจะเห็นได้ว่า การออกแบบ เน้นไปในด้านการตรวจสอบไวยากรณ์เพียงอย่างเดียว โดยไม่สนใจถึงความถูกต้องทางความหมายของภาษา และความผิดพลาดทางด้านไวยากรณ์ที่จะตรวจนั้น อยู่ภายใต้ขอบเขตอันจำกัด ที่ถูกกำหนดไว้ หากพบสิ่งใดที่อยู่ นอกเหนือไปจากขอบเขตที่กำหนดแล้ว ตัวตรวจสอบไวยากรณ์ จะสร้างข่าวสารในลักษณะเป็น ข่าวสารความผิดพลาด เตือนให้ผู้ใช้งานได้ทราบ

รายละเอียดของโปรแกรมการสร้างตัวตรวจสอบไวยากรณ์โปรแกรมภาษาโคบอลจะได้นำเสนอในบทต่อไป



รูปที่ 3-14 แสดงการทำงานต่อเนื่องกันของเฟสที่ 1 (การวิเคราะห์เลกซิเคิล)
และเฟสที่ 2 (การวิเคราะห์ความถูกต้องของไวยากรณ์ภาษา)