บทที่ 3

การสร้างตัวประมวลผลภาษาสอบถามภาษาไทย

ภาษาสอบถามเป็นภาษาระดับสูงประเภทหนึ่ง แต่การที่เราไม่เรียกว่าตัวแปลภาษาสอบถาม เพราะว่าคำสั่งในภาษาสอบถามไม่ได้ถูกแปลเป็นภาษาระดับต่ำหรือภาษาใด ๆ ทั้งสิ้น คำสั่งจะถูกวิเคราะห์และทำงานตามคำสั่งนั้นๆ ทันที ดังนั้นเราจึงเรียกว่าตัวประมวลภาษาสอบถามภาษาไทย(Thai Query Language Processor)

ในบทนี้จะกล่าวถึง รายละเอียดในการสร้างตัวประมวลผลภาษาสอบถามภาษาไทย โดยจะกล่าวถึงรายละเอียดในการออกแบบส่วนต่าง ๆ ของตัวประมวลผล อันได้แก่ หน่วยวิเคราะห์คำ(Lexical Analyzer or Scanner) หน่วยวิเคราะห์วากยสัมพันธ์(Syntax Analyzer or Parser) หน่วยเตรียมการทำงานและหน่วยปฏิบัติตามคำสั่ง

3.1 บทนำ

ถึงแม้หน่วยประมวลผลภาษาสอบถามจะถูกแบ่งออกเป็นหลายหน่วย แต่ลำดับในการทำงานก็ไม่ได้เรียงลำดับเป็นขั้น ๆ ผ่านไปตามแต่ละหน่วย แต่ทั้งนี้การแบ่งออกเป็นหน่วยย่อยๆ ก็เพื่อประโยชน์ในการแบ่งแยกหน้าที่การทำงานซึ่งแตกต่างกันออกจากกัน ให้เด่นชัด เพื่อความสะดวกในการออกแบบและการวิเคราะห์การทำงานของโปรแกรม ตัวอย่างเช่นระหว่างหน่วยวิเคราะห์คำกับหน่วยวิเคราะห์วากยสัมพันธ์นั้น สำหรับในวิทยานิพนธ์นี้นั้น หน่วยวิเคราะห์คำไม่ได้ทำการวิเคราะห์คำจนหมดทั้งคำสั่งแล้วจึงส่งผ่านผลลัพธ์ไปยังหน่วยวิเคราะห์วากยสัมพันธ์ แต่หน่วยวิเคราะห์วากยสัมพันธ์จะทำงานไปพร้อมกับหน่วยวิเคราะห์คำ โดยเวลาใดที่ต้องการรู้ชนิดของคำก็จะเรียกหน่วยวิเคราะห์คำให้วิเคราะห์ให้

ตัวประมวลผลภาษาสอบถามแบ่งออกเป็นหน่วยย่อยดังนี้

1.หน่วยวิเคราะห์คำ(Lexical Analyzer or Scanner) เนื่องจากคำสั่งสอบถามแบ่งเป็น 2 ชนิด ดังนั้นหน่วยวิเคราะห์คำสำหรับคำสั่ง 2 ชนิดนี้ก็แยกจากกันด้วย หน่วยวิเคราะห์คำทำหน้าที่ในการวิเคราะห์คำในประโยคว่าเป็นคำชนิดไหน แล้วแปลงให้อยู่ในรูปของรหัสหรือกลุ่มตัวอักขระที่เหมาะกับการทำงานของหน่วยวิเคราะห์วากยสัมพันธ์ เราเรียกรหัสหรือกลุ่มตัวอักขระนี้ว่าโทคเคน(Token)

2. หน่วยวิเคราะห์วากยสัมพันธ์(Syntax Analyzer or Parser) หรืออาจเรียกได้อีกอย่างหนึ่งว่าหน่วยวิเคราะห์ประโยค แบ่งเป็นสองตัวเช่นเดียวกับหน่วยวิเคราะห์คำ หน่วยนี้ทำหน้าที่ในการวิเคราะห์ประโยค ว่าถูกต้องตามไวยากรณ์ที่กำหนดไว้หรือไม่ โดยการวิเคราะห์นั้นจะใช้โทคเคนเป็นข้อมูลนำเข้า หน่วยวิเคราะห์วากยสัมพันธ์ของประโยคคำสั่งชนิดกึ่งภาษาไทยธรรมชาตินั้น มีความซับซ้อนกว่าของประโยคคำสั่งชนิดไวยากรณ์อิสระจากเนื้อหา ผลลัพธ์ที่ได้จากการวิเคราะห์ประโยคชนิดกึ่งภาษาไทยธรรมชาติจะอยู่ในรูปของ คำสั่งชนิดไวยากรณ์อิสระจากเนื้อหา ซึ่งจะส่งต่อไปยังหน่วยวิเคราะห์วากยสัมพันธ์ของประโยคชนิดไวยากรณ์อิสระจากเนื้อหาอีกทีหนึ่ง

3. หน่วยเตรียมการทำงาน(Execution Preparation Module) เมื่อหน่วยวิเคราะห์วากยสัมพันธ์วิเคราะห์จนรู้ว่าเป็นคำสั่งอะไร ให้ทำอะไรบ้าง ก็จะจัดเตรียมสิ่งแวดล้อมในการทำงานให้เหมาะกับการทำงานตามคำสั่งนั้นๆ เช่นการจัดค่าพารามิเตอร์หรือตัวแปรต่างๆ สำหรับในวิทยาพนธ์นี้เนื่องจากงานของหน่วยนี้มีไม่มากนัก เวลาสร้างตัวประมวลผลจึงไม่ได้แยกหน่วยนี้ออกมาจากหน่วยปฏิการตามคำสั่ง

4. หน่วยปฏิบัติการตามคำสั่ง(Command Execution Module) ทำหน้าที่ในการทำงานตามคำสั่งแต่ละคำสั่ง

5.หน่วยจัดการอุปกรณ์รับและส่งข้อมูล(Input & Output Manager) ทำหน้าที่เกี่ยวกับการแสดงข้อมูลออกทางจอภาพ เครื่องพิมพ์หรือรับ

ข้อมูลเข้าทางแป้นพิมพ์ รวมถึงการรับข้อมูลทางจอภาพในแบบเต็มจอ (Full Screen Editing)

6.หน่วยจัดการฐานข้อมูล(Data Base Manager) ทำหน้าที่ในการจัดการงานต่างๆ ที่เกี่ยวข้องกับแฟ้มข้อมูล เช่น การสร้างแฟ้ม การเปิดแฟ้ม การค้นข้อมูล การเขียนข้อมูลลงแฟ้ม เป็นต้น หน้าที่นี้ทำโดยโปรแกรมสำเร็จรูป (Btrieve)

3.2 หน่วยวิเคราะห์คำ

3.2.1หน่วยวิเคราะห์คำสำหรับคำสั่งชนิดไวยากรณ์อิสระจากเนื้อหา (Scanner of Context Free Grammar Command) หน่วยนี้จะวิเคราะห์ว่าคำในคำสั่งเป็นคำชนิดต่าง ๆตามที่ได้กล่าวไว้แล้วในหัวข้อ 2.3 ในบทที่ 2 หรือไม่

การค้นหาคำสำคัญ

โปรแกรมวิเคราะห์คำ ทำหน้าที่ในการค้นหาคำสำคัญโดยการค้นจากตารางคำสำคัญ 1 แถวประกอบด้วยช่อง 2 ช่อง ช่องแรกคือตัวชี้ไปยังที่อยู่ของของคำสำคัญนั้นๆ (Keyword Pointer) ช่องที่สองคือ รหัสที่แทนโทคเคนของคำสำคัญนั้นๆ วิธีในการค้นนั้นจะใช้ แบบอินเด็กซีเควนเชียล(ISAM) ดังนั้นจึงต้องจัดกลุ่มคำสำคัญให้คำสำคัญที่ขึ้นต้นด้วยตัวอักขระเดียวกันอยู่ในกลุ่มเดียวกัน เช่น คำว่า 'เลิก','เพิ่ม','เปิดแฟ้ม' จะอยู่ในกลุ่มเดียวกัน และ จะต้องมีตารางดัชนีบอกเลขที่แถวเริ่มต้นของคำในกลุ่มตัวอักขระนั้น เช่นสมมุติให้คำที่ขึ้นต้นด้วย 'เ'เริ่มจากแถวที่ 1 ในตารางคำสำคัญ ดังนั้นเลขที่แถวในตารางดัชนีของกลุ่มคำสำคัญที่ขึ้นต้นด้วย'เ' จะมีค่า 1 เราแสดงรายละเอียดของตารางทั้งสองได้ดังต่อไปนี้

เครื่องหมายที่ใช้

* ที่อยู่หน้าคำสำคัญเป็นการบอกว่าตารางใน ๗.ช่องนั้น ไม่ได้เก็บคำสำคัญเอาไว้โดยตรง แต่เก็บที่อยู่ของคำนั้นเอาไว้แทน

แถวที่	ที่อยู่ของคำสำคัญ	โหดเคน
1	*เลิก	70
2	*เพิ่ม	74
3	*เปิดแฟ้ม	72
4	*แก้ไข	76
5	*แก้ที่ผิด	76
6	*แสดงโครงสร้าง	73

ตารางคำสำคัญ

อักขระเริ่มต้น	เลขที่แถวของกลุ่ม
เ	1
แ	4

ตารางดัชนี

รูปที่ 3.1 ตารางคำสำคัญและตารางดัชนีบอกเลขที่แถวเริ่มต้นของกลุ่ม

โปรแกรมสแกนเนอร์ซึ่งใช้ตารางคำสำคัญนี้คือโปรแกรมย่อยชื่อ stoke ขั้นตอนในการค้นเริ่มต้นโดยการนำตัวอักขระตัวแรกของคำ ที่ต้องการนำไปเปรียบเทียบกับคำในตารางไปเปรียบเทียบอักขระในตารางดัชนี แล้วนำค่าแถวที่ได้ไปดึงคำสำคัญที่เริ่มต้นด้วยอักขระตัวนั้นเปรียบเทียบกับคำที่ต้องการค้นไปทีละคำจนกว่าจะพบ เนื่องจาก stoke ถูกเขียนขึ้นด้วยภาษาซี(C language) ดังนั้นตารางคำสำคัญและตารางดัชนีในภาษาซีเป็นดังนี้คือ

```
struct tok_rec {
        char *keyword;    /*ที่อยู่ของคำสำคัญ*/
        char token;       /*โทคเคน*/
 } tok_tab[];
struct tok_rec *tokp;
```

โปรแกรมที่ 3.1 ตารางคำสำคัญเมื่อกำหนดโดยภาษาซี

```
struct commkey {
        char commch;          /*อักขระตัวแรกของคำสำคัญ*/
        int  commadd;         /*เลขที่แถวในตารางคำสำคัญ*/
}
struct commkey *commp;
```

โปรแกรมที่ 3.2 ตารางดัชนีเมื่อกำหนดโดยภาษาซี

<u>การค้นชื่อเขตข้อมูล</u>

ในการค้นว่าคำในคำสั่ง เป็นชื่อเขตข้อมูลหรือหรือไม่นั้น ทำโดยการนำคำนั้นไปเปรียบเทียบกับชื่อเขตข้อมูลในตารางโครงสร้างของแฟ้มข้อมูล ตารางโครงสร้างเมื่ออยู่รูปของภาษาซีเป็นดังนี้คือ

```
struct fstru  {
  char fnum[4];      /*เลขที่เขตข้อมูล*/
  char fname[16];    /*ชื่อเขตข้อมูล*/
  char ftyp[2];      /*ชนิดของเขตข้อมูล*/
  char flen[4];      /*ความยาวของเขตข้อมูลในเชิงตัวอักขระ*/
  char fdec[2];      /*ทศนิยม*/
  int  fln;          /*ความยาวของเขตข้อมูลในเชิงตัวเลข*/
  int  fdc;          /*ทศนิยมในเชิงตัวเลข*/
  char *fcp;         /*ที่อยู่ของเขตข้อมูล*/
  double   fval; /*ค่าของเขตข้อมูล ในกรณีเขตข้อมูลชนิดตัวเลข*/
} fsttab[24];              /*ตารางนี้มีสมาชิกได้ 24 แถว */
```

โปรแกรมที่ 3.3 ตารางโครงสร้างของแฟ้มข้อมูลซึ่งกำหนดโดยภาษาซี

หน่วยวิเคราะห์คำสำหรับนิพจน์

เนื่องจากนิพจน์มีไวยากรณ์ที่ซับซ้อนกว่า หน่วยอื่นๆของคำสั่ง และประกอบขึ้นด้วยองค์ประกอบหลายประเภท อันได้แก่ ชื่อเขตข้อมูล ค่าคงที่ตัวเลข ค่าคงที่ข้อความ เครื่องหมายดำเนินการ และวงเล็บ ดังนั้นจึงแยกหน่วยวิเคราะห์คำสำหรับนิพจน์ออกจากหน่วยวิเคราะห์คำของคำสั่งอื่นๆ หน่วยวิเคราะห์คำของนิพจน์คือโปรแกรมย่อยชื่อ get_token

ผลลัพธ์ที่เกิดจากพาสเซอร์ของนิพจน์(Output from Expression Parser)

จากนิพจน์รูปแบบปกติที่ผู้ใช้งานพิมพ์เข้าสู่เครื่อง พาสเซอร์จะทำให้อยู่ในรูปของ โปลิสโนเตชัน(Polish Notation) และตัวถูกกระทำจะเปลี่ยนให้อยู่ในรูปของโทคเคน ดังตัวอย่างต่อไปนี้

นิพจน์รูปปกติ = อายุ มากกว่า 20 และ ชื่อ = 'ก'

สมมุติว่า ชื่อ คือเขตข้อมูลซึ่งมีเลขที่ของเขตข้อมูลเป็น 1 และอายุมีเลขที่เป็น 4 ดังนั้นจะได้

โปลิสโนเตชัน = F04 I01 > F01 I02 = &

โดยที่โทคเคน F04 หมายถึง อายุ
I01 หมายถึง 20
> หมายถึง มากกว่า
F01 หมายถึง ชื่อ
I02 หมายถึง 'ก'
& หมายถึง และ

ตารางเก็บค่าคงที่

ตัวถูกกระทำมีอยู่สองชนิดใหญ่ๆ คือเขตข้อมูลและค่าคงที่ จากโปลิสโน เตชันจะเห็นได้ว่าค่าคงที่จะถูกเปลี่ยนให้อยู่ในรูปของโทคเคน In โดย n คือ เลขจำนวนเต็มสองหลัก ไม่ได้มีการเก็บค่าของมันเอาไว้ในโปลิสโนเตชันด้วย สแกนเนอร์จะนำค่าของค่าคงที่ไปเก็บเอาไว้ในตารางค่าคงที่ ในการนำค่าของค่าคงที่ไปใช้นั้น จะใช้ลำดับที่ของค่าคงที่เป็นดัชนีในการค้นค่าจากตาราง ตารางค่าคงที่มีรายละเอียดดังนี้

-เลขที่ของค่าคงที่นั้น

-ชนิดข้อมูลของค่าคงที่

-ที่อยู่ของค่าคงที่ กรณีค่าคงที่ข้อความค่านี้จะชี้ไปยังที่เก็บข้อความ

-ค่าทางตัวเลขของค่าคงที่ กรณีค่าคงที่เป็นชนิดตัวเลข

เลขที่	ชนิด	ที่อยู่ของข้อความ	ค่าของค่าคงที่

รูปที่ 3.2 ตารางค่าคงที่

ตารางค่าคงที่เมื่อกำหนดโดยภาษาซี เป็นดังนี้คือ

```
struct id_rec {
  char id_no[3]; /*เลขที่ของค่าคงที่*/
  char id_dtyp;  /*ชนิดข้อมูลของค่าคงที่*/
  char *id_add;  /*ที่อยู่ของค่าคงที่ข้อความ ชี้ไปที่ id_buff*/
  double id*_val;/*ค่าของค่าคงที่ตัวเลข*/
} id_tab[50];    /*เก็บค่าคงที่ได้เพียง 50 ตัว */
struct id_rec *idp;
char id_buff[1024];
```

โปรแกรมที่ 3.4 การกำหนดตารางค่าคงที่โดยภาษาซี

บัฟเฟอร์สำหรับเก็บค่าคงที่ข้อความ

เนื่องจากข้อความมีความยาวไม่แน่นอน ดังนั้นจึงเก็บเฉพาะที่อยู่ของมันเอาไว้ในตารางค่าคงที่ ส่วนข้อความจะนำไปเก็บไว้ในบัฟเฟอร์ชื่อ id_buff ซึ่งมีขนาด 1024 ตัวอักขระ ดังนั้นความยาวรวมของข้อความทุกข้อความจะไม่เกินค่านี้ ถ้าหากต้องการเก็บให้ได้มากกว่านี้และไม่ต้องจองที่เอาไว้ก่อน สามารถทำได้โดยการขอเนื้อที่ในหน่วยความจำแบบพลวัต (Dynamic Allocation)

โทคเคนของคำสำคัญ

คำสำคัญแต่ละคำตลอดจนตัวดำเนินการจะถูกเปลี่ยนให้อยู่ในรูปของโทคเคนดังนี้

คำสำคัญ	โทคเคน	คำสำคัญ	โทคเคน
เลิก,เลิกทำงาน	70	สำเนา,สำเนาไป	79
สร้าง,สร้างแฟ้ม	71	ทำรายงาน,รายงาน	80
เปิดแฟ้ม,ใช้แฟ้ม	72	ไป	81
โครงสร้าง,แสดงโครงสร้าง	73	ปิดแฟ้ม	82
เพิ่ม,ใส่ข้อมูล	74	ปรากฏ	83
!แสดง	75	ลบจอ,ล้างจอ	84
แก้ไข	76	ช่วยด้วย	86
ลบ	77	ถัดไป,ถัด	102
ลบแฟ้ม	78	สำหรับ	104

ตารางที่ 3.1 ตารางแสดงค่าโทคเคนของคำสำคัญ

ตัวดำเนินการ	โทคเคน
ไม่,NOT	33
และ,AND	34
หรือ,OR	35
$	36
*	42
+	43
-	45
/	47

ตัวดำเนินการ	โทคเคน
NEG	46
^,**	41
น้อยกว่า,<	60
เท่ากัน,เท่ากับ,=	61
มากกว่า,>	62
ไม่เท่ากัน,!=,<>	63
<=	64
>=	65

ตารางที่ 3.2 ตารางแสดงค่าโทคเคนของตัวดำเนินการ

3.2.2 หน่วยวิเคราะห์คำสำหรับคำสั่งชนิดกึ่งภาษาไทยธรรมชาติ

หน่วยนี้ทำหน้าที่ในการวิเคราะห์ว่าเป็นคำชนิดไหน ในคำประเภทต่างๆที่ใช้ในคำสั่งชนิดนี้ นอกจากคำเฉพาะแล้ว คำประเภทอื่นๆนั้นหน่วยวิเคราะห์คำของคำเหล่านั้นคือ stoke ซึ่งเป็นโปรแกรมเดียวกับที่ใช้หาคำสำคัญ

โทคเคนของซึ่งแทนชนิดของคำ

คำชนิด	โทคเคน
คำเชื่อมชนิดที่ 1	122
คำสรรพนามบุรุษที่หนึ่ง	123
คำกริยาสกรรมชนิดที่ 1	124
คำสรรพนามใช้ถาม	125

ตารางบันทึกตำแหน่งของคำเฉพาะ

โปรแกรมย่อยที่ทำหน้าที่ในการค้นหาคำเฉพาะ ในประโยคคำสั่งคือ sdatval เมื่อ sdatval รู้ว่ามีคำเฉพาะอยู่ ณ.ตำแหน่งไหนบ้างก็จะบันทึกชนิด ตำแหน่งเริ่มต้น และ ตำแหน่งสิ้นสุดของคำเฉพาะไว้ในตารางบันทึกตำแหน่ง รายละเอียดของตารางนี้เป็นดังนี้คือ

```
struct datt {
char dtyp;      /*ชนิดของคำเฉพาะ(Data value typ*/
char *dst;      /*เก็บตำแหน่งเริ่มต้น*/
char *dend;     /*ตำแหน่งสิ้นสุด*/
}
```

ชนิดของคำเฉพาะ	ตำแหน่งเริ่มต้น	ตำแหน่งสิ้นสุด

รูปที่ 3.3 ตารางบันทึกตำแหน่งของคำเฉพาะ

ตารางบันทึกตำแหน่งของเขตข้อมูล

โปรแกรมย่อย fldscan ทำหน้าที่ในการค้นหาชื่อเขตข้อมูลในประโยคคำสั่งชนิดกึ่งภาษาธรรมชาติ เมื่อพบก็จะบันทึกชนิด ตำแหน่งเริ่มต้นและสิ้นสุด (ตำแหน่งในคำสั่ง) และที่อยู่ของชื่อเขตข้อมูลในตารางโครงสร้างแฟ้มข้อมูล เอาไว้ในตารางบันทึกตำแน่งของเขตข้อมูล มีรายละเอียดดังนี้

```
struct fldt {
  char ft;          /*ชนิดของเขตข้อมูล*/
  char *fst;        /*เก็บตำแหน่งเริ่มต้น*/
  char *fend;       /*ตำแหน่งสิ้นสุด*/
  char    *fp;      /*ที่อยู่ของชื่อเขตข้อมูลในตารางโครงสร้าง*/
    }
```

ชนิด	ตำแหน่งเริ่ม	ตำแหน่งสิ้นสุด	ที่อยู่ของชื่อในตารางโครงสร้างแฟ้ม

รูปที่ 3.4 ตารางบันทึกตำแหน่งของเขตข้อมูล

3.3 พาสเซอร์(Parser)

3.3.1 พาสเซอร์ของคำสั่งชนิดไวยากรณ์อิสระจากเนื้อหา

พาสเซอร์ทำหน้าที่ในการตรวจสอบคำสั่งว่าถูกต้องตามไวยากรณ์หรือไม่ คือจะวิเคราะห์ว่าใช้คำซึ่งถูกเปลี่ยนให้อยู่ในรูปของโทคเคนแล้วและการวางตำแหน่งของคำถูกต้องหรือไม่ การที่พาสเซอร์รู้ว่าถูกต้องหรือไม่ก็เพราะอาศัยข้อกำหนดทางไวยากรณ์ของคำสั่งนั้นๆ

ไวยากรณ์(Grammar) มีส่วนประกอบ 4 ส่วนคือ

1. เทอร์มินอล (Terminals) คือกลุ่มของตัวอักขระหรือ

โทคเคนที่แบ่งแยกต่อไปไม่ได้อีกแล้ว เช่นคำสำคัญ สัญญลักษณ์ที่แทนตัวดำเนินการ เมื่อดูจากกฎในไวยากรณ์ คือกลุ่มตัวอักขระที่อยู่แต่ทางขวามือของกฎเพียงอย่างเดียว จะไม่อยู่ทางซ้ายมือของกฎเลย

2. นอนเทอร์มินอล (Noneterminals) คือกลุ่มของโทคเคนที่ยังแบ่งแยกย่อยต่อไปได้อีก ส่วนย่อยเหล่านั้นอาจเป็นนอนเทอร์มินอลหรือเทอร์มินอลก็ได้ ถ้าดูจากกฎก็หน่วยที่จะต้องอยู่ทางขวามือของกฎอย่างน้อยหนึ่งกฎ

3. สัญญลักษณ์เริ่มต้น (Start Symbol) คือ นอนเทอร์มินอลที่ถูกเลือกเป็นตัวแรกสุดที่จะพิจารณาปริมาณที่ย่อยลงไปอีก

4. กฎที่ใช้เปลี่ยนแปลง (Production or Rewriting Rules) คือกฎที่ใช้กำหนดว่านอนเทอร์มินอลจะประกอบด้วยนอนเทอร์มินอลหรือเทอร์มินอลอะไรบ้าง กฎทุกกฎจะประกอบด้วย นอนเทอร์มินอลอยู่ทางซ้ายมือ ตามด้วยลูกศร และทางขวามือประกอบด้วยเทอร์มินอลหรือนอนเทอร์มินอล

ตัวอย่างของไวยากรณ์เช่น

```
1. EXPRESS      --> TERM EXPRESSES
2. EXPRESSES    --> ASOP TERM EXPRESSES
                --> null
3. TERM         --> FACTOR TERMS
4. TERMS        --> MDOP FACTOR TERMS
                --> null
5. FACTOR       --> '(' EXPRESS ')'
                --> OPERAND
6. ASOP         --> '+'
                --> '-'
7. MDOP         --> '*'
                --> 'div'
```

ในตัวอย่างเป็นไวยากรณ์ของนิพจน์(Expression) ในไวยากรณ์หนึ่งไวยากรณ์จะประกอบขึ้นด้วยกฎหลายกฎ

การตรวจสอบไวยากรณ์หรือที่เรียกว่าการพาส(Parse)สามารถทำได้หลายวิธี โดยจะใช้วิธีไหนได้บ้างนั้น ขึ้นอยู่กับชนิดของไวยากรณ์ที่เราสร้างให้กับพาสเซอร์ด้วย วิธีการพาสที่แพร่หลายได้แก่วิธีการตรวจสอบจากบนลงล่าง (Top-down Parsing) โดยปกติแล้วจะมีการย้อนกลับไปตรวจสอบใหม่(Backtrack) เพราะหลังจากที่ได้เลือกกฎในไวยากรณ์ได้และทำการตรวจสอบปรากฏผลว่ากฎที่เลือกไม่เหมาะกับประโยคที่นำมาตรวจสอบ จึงต้องย้อนกลับมาเลือกกฎใหม่ ถ้าหากไม่สามารถเลือกกฎที่เหมาะสมได้เลย แสดงว่าประโยคที่นำมาตรวจนั้นไม่ถูกต้อง

การย้อนกลับไปตรวจสอบใหม่ทำให้การทำงานช้าและยุ่งยาก จึงได้มีผู้คิดไวยากรณ์แบบแอลแอลวัน(LL(1) Grammar) ซึ่งไม่ต้องมีการย้อนกลับมาตรวจสอบใหม่ ในทางเลือกสองทางของกฎ พาสเซอร์ของไวยากรณ์แบบนี้สามารถเลือกกฎได้ถูกต้องโดยการตรวจสอบเทอร์มินอลตัวต่อไปว่าเหมาะที่จะเลือกใช้กฎไหน คุณสมบัติของไวยากรณ์แบบแอลแอลวันเป็นดังนี้คือ

- ในกรณีที่มีนอนเทอร์มินอลเดียวกันปรากฏอยู่ทางซ้ายมือของกฎมากกว่าหนึ่งกฎ ส่วนที่อยู่ทางขวามือของกฎเหล่านั้นจะต้องขึ้นต้นด้วยเทอร์มินอลคนละตัวกันหรือนอนเทอร์มินอลที่อ้างไปถึงเทอร์มินอลคนละตัวกัน

<u>ตัวอย่างไวยากรณ์ของคำสั่งที่มีไวยากรณ์แบบแอลแอลวัน</u>

เครื่องหมายที่ใช้

{...} โทคเคนหรือสัญญลักษณ์ที่ใช้ตรวจสอบล่วงหน้า(Look Ahead Symbol)

'...' คำสำคัญ เวลาส่งเข้าพาสเซอร์จะอยู่ในรูปของโทคเคน แต่เพื่อความสะดวกในการอ่าน จึงเขียนรูปเต็ม

null หมายถึงไม่มีอักขระ หรือโทคเคนใดๆ

1. <คำสั่ง!แสดง>	-->	'!แสดง' <แสดง2>	{'!แสดง'}
2. <แสดง2>	-->	<แสดง3><แสดง4>	{'ถัด'}
	-->	<แสดง5><แสดง6>	{'สำหรับ'}
	-->	null	{null string}
3. <แสดง3>	-->	'ถัด' <เลข>	{'ถัด'}
4. <แสดง4>	-->	'สำหรับ' <นิพจน์>	{'สำหรับ'}
	-->	null	{null string}
5. <แสดง5>	-->	'สำหรับ' <นิพจน์>	{'สำหรับ'}
6. <แสดง6>	-->	'ถัด' <เลข>	{'ถัด'}
	-->	null	{null string}

สมมุติว่า ผู้ออกคำสั่งได้ออกคำสั่งดังนี้

!แสดง ถัด 20

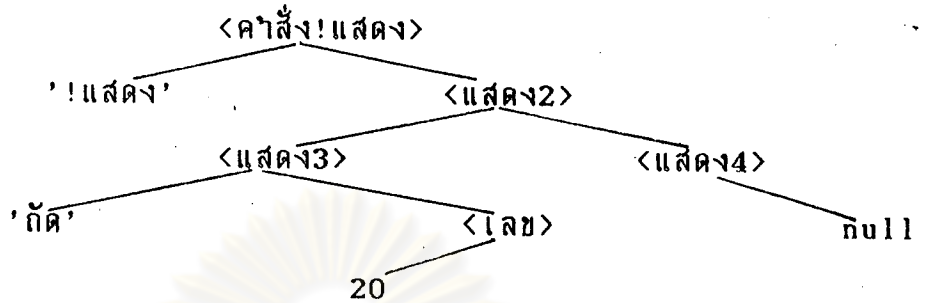
พาสเซอร์ของคำสั่ง!แสดง จะทำงานเป็นไปตามลำดับดังนี้คือ

1. เมื่อพาสเซอร์หลัก(Main Parser)เรียก stoke และได้โทคเคนของ'!แสดง' ก็จะส่งการทำงานต่อไปยัง พาสเซอร์<คำสั่ง!แสดง>

2. พาสเซอร์<คำสั่ง!แสดง> เรียกต่อไปยังพาสเซอร์<แสดง2>

3. พาสเซอร์<แสดง2> เรียก stoke ได้โทคเคนตัวต่อไปคือ 'ถัด' ดังนั้น จึงเรียกพาสเซอร์<แสดง3> ซึ่งทำหน้าจัดการเกี่ยวกับ 'ถัด' จากนั้น พาสเซอร์<แสดง2>เรียก พาสเซอร์<แสดง4>

การพาสดังที่ยกมาเป็นตัวอย่างคือการสร้างพาสทรี (Parse Tree) ของคำสั่งนั้นนั่นเอง พาสทรีที่ได้เป็นดังนี้คือ



รูปที่ 3.5 พาสทรีของคำสั่ง !แสดง ถัด 20

ในระหว่างการพาสแต่ละขั้นตอนถ้าเกิดความผิดพลาด(คำสั่งไม่เป็นไปตามไวยากรณ์)ขึ้น พาสเซอร์จะทำให้(Set)ค่าของแฟลกข้อผิดพลาด (Error Flag) ให้มีค่าตามประเภท ของความผิดพลาดนั้น

การที่เรียกไวยากรณ์แบบนี้ว่าแอลแอลวัน ก็เพราะการพาสทำจากซ้ายไปขวา(Left to Right = แอลตัวแรก) และเลือกกฎที่เหมาะสมโดยการตรวจสอบโทคเคน(เทอร์มินอล)ตัวซ้ายสุด(Leftmost = แอลตัวที่สอง)

ไวยากรณ์ของภาษาสอบถามภาษาไทยชนิดไวยากรณ์อิสระจากเนื้อหานี้แสดงไว้ในภาคผนวกในรูปแบบทางวากยสัมพันธ์ และรูปแบบแบคลัส-นอร์(BNF)

ถ้าหากเป็นตัวแปลภาษา(Compiler)แล้วขั้นตอนหนึ่งที่จะขาดไปไม่ได้ก็คือการแปลจากภาษาสูงไปเป็นภาษาต่ำ วิธีการในการสร้างภาษาต่ำในขั้นตอนเดียวกับการวิเคราะห์วากยสัมพันธ์(การพาส) ในเรื่องการแปลภาษาจะเรียกว่า<u>การแปลโดยอาศัยวากยสัมพันธ์</u> (Syntax-directed Translation) แต่เนื่องจากเป็นภาษาสอบถามจึงไม่มีการแปลไปเป็นภาษาระดับต่ำ เพียงแต่มีการจัดรูปแบบคำสั่งเสียใหม่ หรือการจัดเตรียมการทำงาน ตัวอย่างที่เห็นได้ชัดได้แก่ การพาสนิพจน์ โดยโปรแกรมชื่อ exp_par ในขณะที่พาสนิพจน์ไปด้วยนั้น ก็จะมีการเรียกโปแกรมชื่อ code ทำหน้าที่ในการเปลี่ยนนิพจน์ให้อยู่ในรูปโพลิสโนเตชันไปด้วย

3.3.2 โครงสร้างข้อมูล(Data Structure)ที่ใช้สำหรับพาสเซอร์ของคำสั่งชนิดไวยากรณ์อิสระจากเนื้อหา

โครงสร้างข้อมูลหมายถึงตาราง หรือสแตค(Stack)หรือบัฟเฟอร์ต่าง ๆ ที่ใช้สำหรับการทำงานของพาสเซอร์ มีดังต่อไปนี้คือ

สแตคสำหรับตัวดำเนินการ (Operator Stack)

เมื่อพาสเซอร์ของนิพจน์ พาสไปพบตัวดำเนินการก็จะเก็บ (Push) ตัวดำเนินการเอาไว้ในสแตคนี้ เมื่อถึงขั้นตอนที่เหมาะสมก็จะดึง(Pop) ออกมาจากสแตคเพื่อนำไปสร้างโพลิสโนเตชัน การเก็บหรือดึงตัวดำเนินการเป็นไปตามไวยากรณ์ของนิพจน์

สแตคสำหรับตัวถูกกระทำ(Operand Stack)

เมื่อพาสเซอร์ของนิพจน์ พาสไปพบตัวถูกกระทำก็จะนำมาเก็บไว้ในสแตคนี้

บัฟเฟอร์สำหรับเก็บโพลิสโนเตชัน(Polish Notation Buffer)

ใช้สำหรับเก็บโพลิสโนเตชันซึ่งเป็นผลลัพธ์ที่ได้จาก exp_par บัฟเฟอร์นี้คือ posfixb ซึ่งมีขนาด 256 ตัวอักขระ

3.3.3 การวิเคราะห์ไวยากรณ์ของคำสั่งชนิดกึ่งภาษาไทยธรรมชาติ

การวิเคราะห์ไวยากรณ์และการแปลง (Translation) คำสั่งชนิดนี้ ทำโดยโปรแกรมชื่อ csent

ขั้นตอนการวิเคราะห์ไวยากรณ์ของคำสั่งชนิดนี้

1. ในการวิเคราะห์นี้ จะสนใจเฉพาะหน่วยคุณสมบัติเท่านั้น

2. เริ่มโดยการค้นหาคำเฉพาะ ทุกคำที่มีอยู่ในคำสั่งและบันทึก ชนิด จุดตั้งต้น จุดสิ้นสุดเอาไว้ในตารางบันทึกตำแหน่งของคำเฉพาะ เมื่อค้นหาคำเฉพาะได้จนหมดแล้วจะใส่อักขระสิ้นสุด (Null-terminated) เอาไว้ท้ายคำเฉพาะคำสุดท้าย เพราะจะไม่สนใจคำสั่งในส่วนที่เหลือ ถ้าไม่พบคำเฉพาะถือว่าคำสั่งไม่ถูกต้องจะเลิกทำงาน ถ้าพบไปทำต่อข้อ 3 (หน้าที่ในการค้นคำเฉพาะทำโดยโปรแกรมชื่อ sdatval())

3. จัดค่าแฟลกและตัวแปลต่าง ๆ ให้มีค่าเริ่มต้น(Initialize)

4. ค้นหาคำเชื่อมชนิดที่1 ถ้าพบจะตัดส่วนของประโยคที่อยู่ด้านหน้าทิ้งไป รวมทั้งคำชนิดนี้ด้วย งานนี้ทำโดยโปรแกรมชื่อ conj ถ้าไม่พบคำเชื่อมชนิดนี้ไปทำข้อ 5 ถ้าพบไปทำข้อ 8

5. ค้นหาคำสรรพนามบุรุษที่หนึ่ง ถ้าพบตัดส่วนของประโยคที่อยู่ด้านหน้าทิ้งไปรวมทั้งคำชนิดนี้ด้วย งานนี้ทำโดยโปรแกรม pronoun

6. ค้าหาคำกริยาสกรรมชนิดที่1 ถ้าพบตัดประโยคด้านหน้าทิ้งไป งานนี้ทำโดยโปรแกรม tv1

7. หาคำสรรพนามใช้ถามทุกคำที่มีอยู่ ตัดส่วนของประโยคด้านหน้าทิ้งไป รวมคำสรรพนามใช้ถามตัวสุดท้ายด้วย งานนี้ทำโดยโปรแกรม itg

8. ส่วนต่อจากนี้ไปเป็นการวิเคราะห์หน่วยคุณสมบัติ เริ่มโดยการค้นหา(scan)เขตข้อมูลทั้งหมดที่มีอยู่ในคำสั่งและทำการบันทึกชนิด จุดตั้งต้นและจุดสิ้นสุดเอาไว้ใน ตารางบันทึกตำแหน่งของชื่อเขตข้อมูล งานในข้อนี้เป็นหน้าที่ของfldscan

9. เริ่มหาว่าเขตข้อมูลไหนที่อยู่ใกล้คำเฉพาะมากที่สุด โดยการตรวจค้นจากตารางบันทึกตำแหน่งของชื่อเขตข้อมูล โดยหาเขตข้อมูลที่มีจุดสิ้นสุด(fend) มีค่าใกล้จุดตั้งต้นของคำเฉพาะมากที่สุด งานเหล่านี้ทำโดย fseach ในกรณีที่ไม่มีเขตข้อมูลที่อยู่หน้าคำเฉพาะเลย ให้ถามผู้ออกคำสั่งว่าคำเฉพาะนี้หมายถึงเขตข้อมูลไหน โดยการเรียก askf เมื่อได้ชื่อเขตข้อมูลมาแล้วก็ย้ายเข้าไปไว้ในบัฟเฟอร์ ชื่อ tbuff

10. ค้นหาคำปฏิเสธ 'ไม่' ถ้าพบให้เซทแฟลก negf ให้มีค่าเป็น 1 งานนี้ทำโดย โปรแกรมชื่อ negs

11. ค้นหาคำกริยาเปรียบเทียบ ทำการค้นไม่เกินจุดตั้งต้นของคำเฉพาะ โทคเคนของคำกริยาเหล่านี้คือตัวดำเนินการ(Operator) กรณีไม่พบให้จัดตัวดำเนินการที่เหมาะสมให้ โดยดูจากชนิดของคำเฉพาะ ถ้าเป็นชนิดอักขระให้ใช้ตัวดำเนินการ '$' ถ้าเป็นตัวเลขใช้ '=' และถ้า negf มีค่าเป็น 1 จะใช้ตัวดำเนินการที่เป็นนิเสธแทน เมื่อได้ตัวดำเนินการมาเรียบร้อยให้นำไปไว้ในบัฟเฟอร์ งานทั้งหมดนี้ทำโดย comps

12. ย้ายคำเฉพาะไปไว้ในบัฟเฟอร์

13. ถ้าคำเฉพาะยังไม่หมดบันทึกชนิดของคำเฉพาะคำเก่าไว้ในตัวแปร odtyp และ ค้นหาคำเชื่อมชนิดที่2 โดยโปรแกรม logs ถ้าไม่พบให้ใช้คำเชื่อม 'และ' และย้ายคำเชื่อมที่ได้ไปไว้ในบัฟเฟอร์ ถ้าคำเฉพาะหมดแล้วไปที่ข้อ 15

14. ทำการเลือกเขตข้อมูลที่เหมาะสมโดยโปรแกรม fseach แล้วนกลับไปที่ข้อ 10

15. ทำการย้ายคำสั่งในบัฟเฟอร์ซึ่งอยู่ในรูปคำสั่งชนิดไวยากรณ์ไม่ขึ้นกับเนื้อหาไปไว้ในบัฟเฟอร์ของคำสั่งชนิดไวยากรณ์ไม่ขึ้นกับเนื้อหาชื่อ qbuff

16. เรียกโปรแกรม list ซึ่งทำหน้าที่วิเคราะห์และทำงานตามคำสั่ง '!แสดง'

17. จบการทำงาน

ขั้นตอนการทำงานของcsentแสดงได้ด้วยผังงานดังนี้

<u>เครื่องหมายที่ใช้</u>

<ฉ> = คำเฉพาะ

<สน1> = คำสรรพนามบุรุษที่หนึ่ง

<ช1> = คำเชื่อมชนิดที่หนึ่ง

<ส1> = คำสกรรมกริยาชนิดที่1

<สน.ถาม> = คำสรรพนามใช้ถาม

<กป> = คำกริยาเปรียบเทียบ

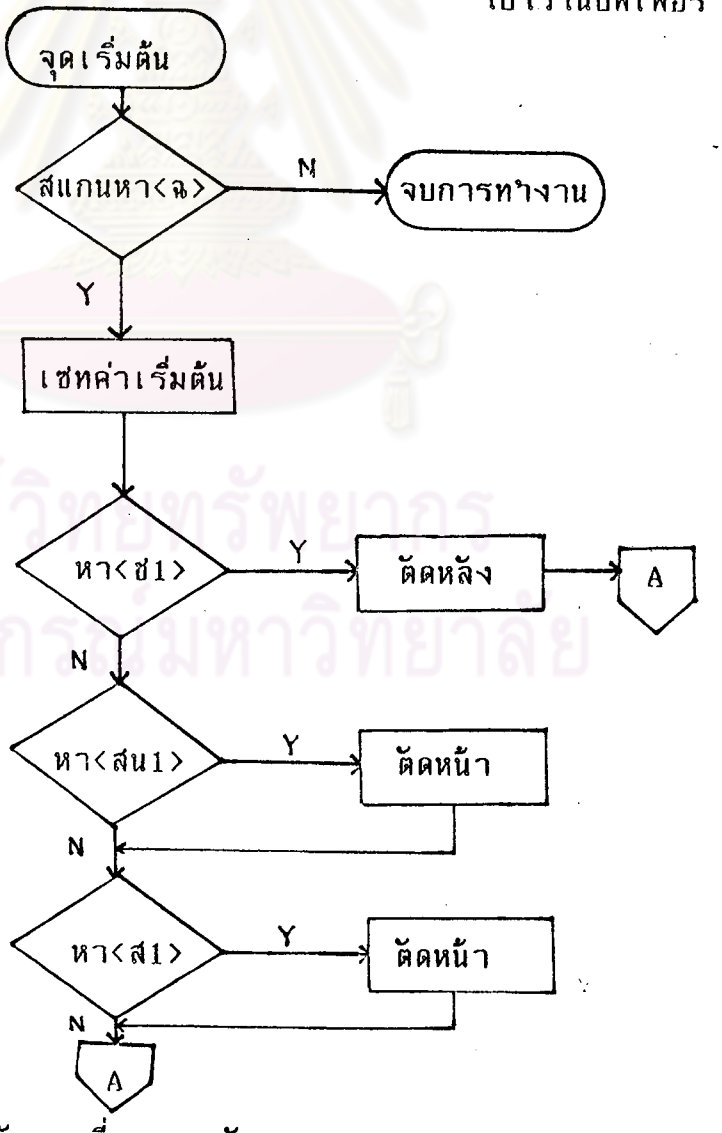
<ช2> = คำเชื่อมชนิดที่2

ตัดหลัง = ตัดส่วนหลังของประโยค

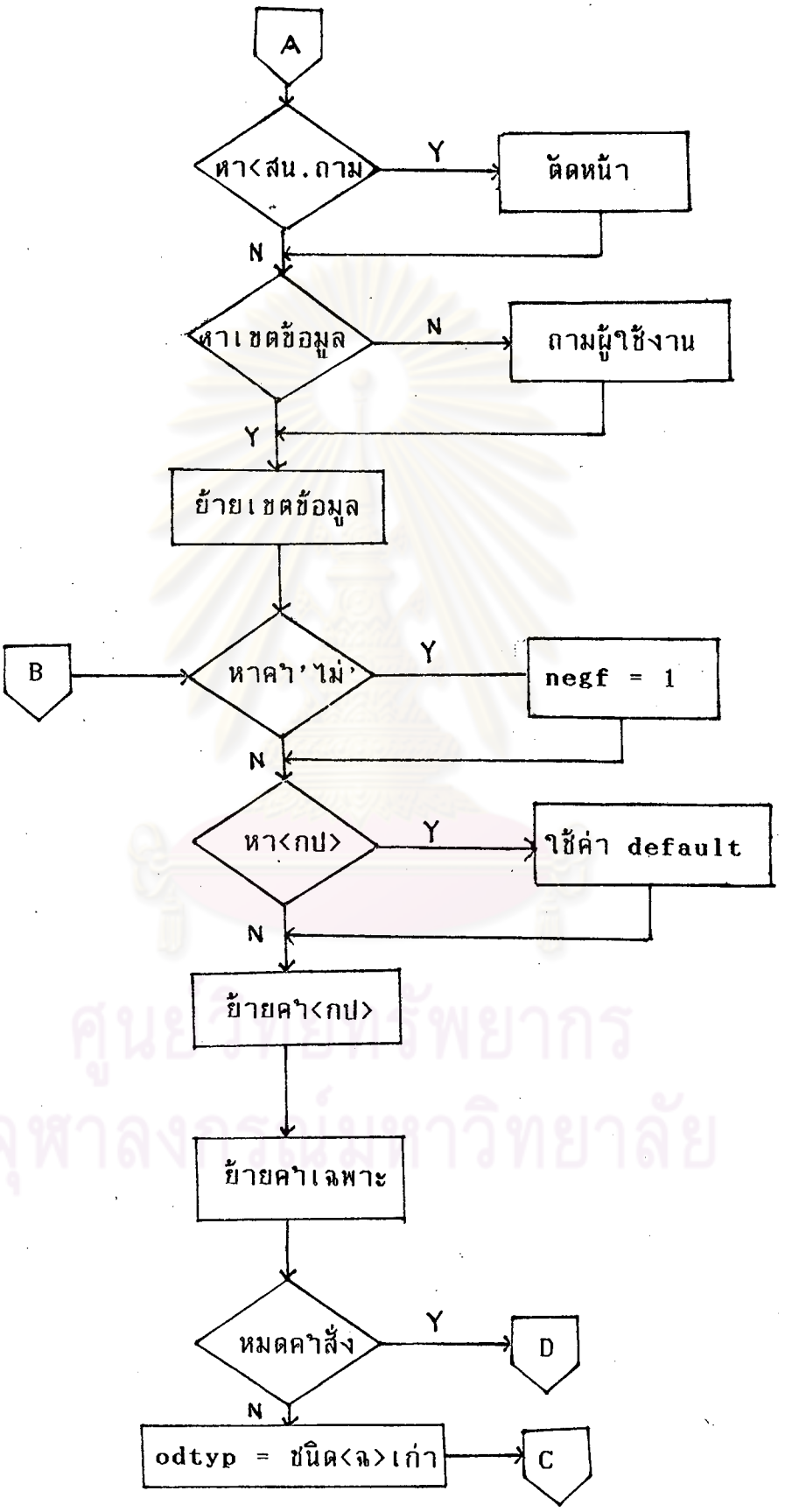
ตัดหน้า = ตัดส่วนหน้าของประโยค

หาเซตข้อมูล = หาเซตข้อมูลที่อยู่หน้า<ฉ>

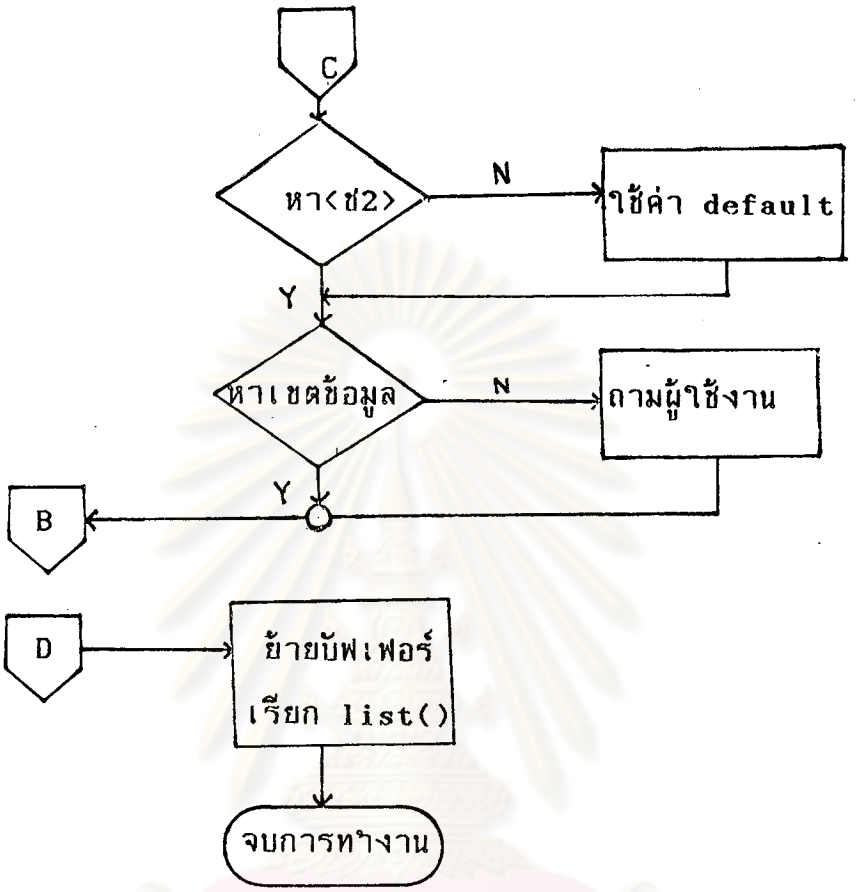
ย้ายเซตข้อมูล = ย้าย(move)เซตข้อมูลไปไว้ในบัฟเฟอร์



ผังงานที่ 3.1 ผังการทำงานของ csent



ผังงานที่ 3.1 ผังการทำงานของ csent(ต่อ)



ผังงานที่ 3.1 ผังการทำงานของ csent (ต่อ)

3.3.4 หน่วยปฏิบัติการตามคำสั่ง

ภายหลังจากที่พาสเซอร์หลัก (Main Parser) ซึ่งคือโปรแกรมหลักชื่อ main ทำการค้นหาคำสำคัญโดยใช้ stoke จะได้โทคเคนของคำสำคัญตัวแรกสุดของคำสั่ง จากนั้นนำโทคเคนที่ได้ไปใช้ในการเรียกหน่วยปฏิบัติการตามคำสั่ง โดยเรียกโปรแกรมชื่อ go ทำหน้าที่ในการเรียกหน่วยปฏิบัติการตามคำสั่ง ถ้าไม่พบคำสำคัญ ก็จะไปเรียกหน่วยปฏิบัติการตามคำสั่งของคำสั่งชนิดกึ่งภาษาธรรมชาติซึ่งคือ csent

ในหน่วยปฏิบัติการตามคำสั่งแต่ละหน่วยนั้น จะทำหน้าที่ในการพาสคำสั่งส่วนที่เหลือ และทำงานตามคำสั่งนั้นด้วย

ต่อไปนี้เป็น ตารางแสดงให้เห็นว่าคำสั่งอะไร ทำงานโดยใช้หน่วยปฏิบัติการชื่ออะไร

คำสั่ง	หน่วยปฏิบัติการตามคำสั่ง
เลิก	quit
สร้าง	create
เปิดแฟ้ม	use
เพิ่ม	append
!แสดง	list
แก้ไข	edit
ลบ	del
ลบแฟ้ม	delf
สำเนา	copy
รายงาน	report
ไป	go_to
ปิดแฟ้ม	close_fil
ลบจอ	clear
ช่วยด้วย	help

ตาราง 3.3 ตารางแสดงหน่วยปฏิบัติการ

3.3.5 หน่วยจัดการอุปกรณ์รับและส่งข้อมูล

หน้าที่ของงานในหน่วยนี้เกี่ยวข้องกับการรับและส่งข้อมูลออกทางอุปกรณ์รับส่งข้อมูล ได้แก่แป้นพิมพ์ จอภาพ และเครื่องพิมพ์ แบ่งงานหน่วยย่อยๆได้ดังนี้

1.การรับและแก้ไขข้อมูลแบบรับทีละบรรทัด(Line Input) โดยอาศัยจอภาพและแป้นพิมพ์

2.การรับ การแก้ไขข้อมูลในลักษณะเต็มจอภาพ (Full-screen Input) โดยเราสามารถเลื่อนเคอเซอร์ไปยังตำแหน่งที่เราต้องการได้ทุกทิศทาง

3.การแสดงผลออกทางเครื่องพิมพ์

4.การลบจอภาพ

ในการรับข้อมูลที่เป็นภาษาไทยทั้งแบบทีละบรรทัด และแบบเต็มจอภาพนั้น เนื่องจากการเก็บอักขระไทยเอาไว้ในบัฟเฟอร์ใช้วิธีการเก็บแบบอยู่ในบรรทัดเดียวกันหมด ถ้าเราไม่มีไวยากรณ์ในการบังคับการเก็บลำดับตัวอักขระไว้ในบัฟเฟอร์หรือในจานแม่เหล็กแล้ว อาจส่งผลทำให้เกิดความผิดพลาดในการนำข้อมูลไปใช้งานต่อไป

ไวยากรณ์สำหรับการจัดลำดับตัวอักขระไทยที่รับจากแป้นพิมพ์

เนื่องจากตัวอักขระไทยประกอบด้วยกลุ่มตัวอักขระ 3 ชนิด ได้แก่ พยัญชนะ สระ และ วรรณยุกต์ และการเก็บตัวอักขระไว้ในหน่วยความจำของเครื่องหรือเก็บไว้ในจานแม่เหล็ก เก็บอยู่ในบรรทัดเดียวกันหมด ดังนั้นลำดับในการเก็บตัวอักขระ จะส่งผลกระทบต่อการนำข้อมูลที่เป็นตัวอักขระไปใช้งานเกี่ยวกับ การค้นหรือการเปรียบเทียบตัวอักขระ เราแบ่งตัวอักขระตามชนิดและตำแหน่งดังนี้คือ

1.กลุ่มตัวอักขระที่เป็นวรรณยุกต์

2.กลุ่มตัวอักขระที่เขียนอยู่ในบรรทัดเดียวกันกับพยัญชนะ ได้แก่ พยัญชนะ สระบางตัว (เ แ ไ ไ) และตัวอักขระทุกตัวในภาษาอังกฤษ

3.กลุ่มตัวอักขระที่เป็นสระซึ่งอยู่ส่วนล่างของพยัญชนะ

4.กลุ่มตัวอักขระที่เป็นสระซึ่งอยู่ส่วนบนของพยัญชนะ

ในการรับตัวอักขระชนิดต่างๆมาจากแป้นพิมพ์ ถ้าหากลำดับตัวอักขระที่รับเข้ามาไม่ถูกต้อง หลักในการแก้ไขปัญหามีอยู่สองทางคือไม่ยอมรับตัวอักขระที่ลำดับไม่ถูกต้อง หรือแก้ไขลำดับให้ถูกต้องถ้าทำได้ ถ้าทำไม่ได้ก็ไม่ยอมรับ สำหรับในวิทยานิพนธ์นี้จะใช้วิธีการแรกคือไม่ยอมรับตัวอักขระที่ผิดลำดับ

<u>ไวยากรณ์ลำดับของตัวอักขระที่ยอมรับได้มีดังนี้</u>

1. <ลำดับ>	-->	<ต้น><ตาม>	{ <2> }
2. <ต้น>	-->	<2>	{ <2> }
3. <ตาม>	-->	<ตาม1>	{ <1> ; <2> }
	-->	<ตาม2><ตาม1>	{ <3> ; <4> }
4. <ตาม1>	-->	<1>	{ <1> }
	-->	<null>	{ <2> }
5. <ตาม2>	-->	<3>	{ <3> }
	-->	<4>	{ <4> }
6. <1>	-->	ตัวอักขระประเภทที่ 1	
7. <2>	-->	ตัวอักขระประเภทที่ 2	
8. <3>	-->	ตัวอักขระประเภทที่ 3	
9. <4>	-->	ตัวอักขระประเภทที่ 4	

ตัวอย่างลำดับตัวอักขระที่ถูกต้อง

ค ◌ น

ถ ◌ ก

ท ◌ ่

ส ◌ ้

ตัวอย่างลำดับตัวอักขระที่ไม่ถูกต้อง

ท ่◌

ส ้◌

กฎของตัวอักขระไทยที่เกี่ยวข้องกับโปรแกรมบรรณาธิกรภาษาไทย

งานแก้ไขหรือรับข้อมูลจากจอภาพนับเป็นงานสำคัญส่วนหนึ่งที่เกี่ยวข้องกับงานจัดการฐานข้อมูล โปรแกรมที่ทำหน้าที่นี้คือโปรแกรมบรรณาธิกร(Editor) ซึ่งทำหน้าที่รับและแก้ไขข้อมูลในแบบเต็มจอ โปรแกรมบรรณาธิกรทำงานหลักอยู่สองอย่างคือการเลื่อนเคอเซอร์ (Cursor)และการรับ การแก้ไขการลบข้อมูล เนื่องจากการทำงานเกี่ยวข้องกับตัวอักขระไทย เพื่อความสะดวกในการเขียนโปรแกรมจึงได้กำหนดกฎเกี่ยวกับตัวอักขระที่เกี่ยวข้องกับโปรแกรมบรรณาธิกรดังนี้

1.ชุดตัวอักขระหรือรหัสพิเศษที่อยู่ทางซ้ายมือของเคอเซอร์เราเรียกว่าไวยากรณ์ทางซ้าย(Left Grammar = Lgram) แบ่งออกเป็น 7 ชนิดดังนี้

1.<L1> --> รหัสที่บอกว่าเป็นจุดตั้งต้นของตัวอักขระ(Top of Text)

2.<L2> --> <2><1>

3.<L3> --> <2>

4.<L4> --> <3>

5.<L5> --> <4>

6.<L6> --> <3><1>

7.<L7> --> <4><1>

ไวยากรณ์ทางซ้ายใช้ประโยชน์สำหรับ การเลื่อนเคอเซอร์ไปทางซ้าย การลบตัวอักขระโดยลบจากอักขระทางซ้ายของเคอเซอร์ การแทรกหรือการเพิ่มตัวอักขระ

2.ชุดตัวอักขระหรือรหัสพิเศษที่นับจากเคอเซอร์ไปทางขวามือ เราเรียกว่าไวยากรณ์ทางขวา(Right Grammar) แบ่งออกเป็น 7 ชนิด ดังนี้

1.<R1> --> รหัสที่บอกว่าเป็นจุดสิ้นสุดข้อความ(End of Text)
2.<R2> --> <2><1>
3.<R3> --> <2><2>
4.<R4> --> <2><3><2>
5.<R5> --> <2><4><2>
6.<R6> --> <2><3><1>
7.<R7> --> <2><4><1>

ไวยากรณ์ทางขวาใช้ประโยชน์สำหรับการเลื่อนเคอเซอร์ไปทางขวา การลบตัวอักขระที่อยู่ ณ.ตำแหน่งของเคอเซอร์

3.ในกรณีการเพิ่มตัวอักขระ จากด้านท้ายสุดของข้อความ (End of Text) หรือการแทรกตัวอักขระ ชุดตัวอักขระที่เกิดขึ้นคือ ไวยากรณ์ทางซ้ายและชนิดของตัวอักขระที่รับเข้ามาใหม่เราจะเรียกชุดอักขระชนิดนี้ว่าไวยากรณ์เพิ่มข้อมูล(Data entry grammar) เป็นดังนี้

1.<D1> --> <L1><2>

2.<D2> --> <L3><1> ! <L3><4> ! <L4><1>

3.<D3> --> <L2><2> ! <L3><2> ! <L4><2> ! <L5><2> !
<L6><2> ! <L7><2>

4.<D4> --> <L3><3>

5.<D5> --> <L5><1>
