บทที่ 2

อินเตอร์พริเตอร์ภาษาเบสิก

อินเตอร์พริเตอร์ เป็นตัวแปลภาษาแบบหนึ่ง ซึ่งในการแปลจะทำการแปลทีละบรรทัดหรือคำสั่ง และปฏิบัติการโดยทันทีไปทีละคำสั่งจนกว่าจะหมดคำสั่งในโปรแกรมนั้น ลักษณะนี้แตกต่างไปจากคอมไพเลอร์ (COMPILER) ซึ่งแปลคำสั่งในโปรแกรมเป็นภาษาเครื่องเสียก่อนแล้วจึงปฏิบัติการตามคำสั่งจากภาษาเครื่องของโปรแกรมนั้น ซึ่งเป็นข้อแตกต่างที่เห็นได้ชัดมาก นอกจากนี้การใช้ภาษาที่ใช้อินเตอร์พริเตอร์ยังสามารถแก้ไขเปลี่ยนแปลงโปรแกรมได้ง่ายซึ่งประหยัดเวลา**กว่าคอมไพเลอร์ที่ต้องทำ**การแปลโปรแกรมหมดทั้งโปรแกรมเสียก่อนจึงทราบข้อผิดพลาดของโปรแกรมนั้น ๆ

ในการปรับปรุงแก้ไขอินเตอร์พริเตอร์เพื่อเพิ่มคำสั่งต่าง ๆ นั้น ต้องเข้าใจในระบบการทำงานพื้นฐานของอินเตอร์พริเตอร์ในส่วนต่าง ๆ ดังนี้ แบบการทำงาน, โครงสร้างของข้อมูล, การแบ่งหน่วยความจำหลักและตาราง (TABLE) ต่าง ๆ ในอินเตอร์พริเตอร์รวมทั้งโครงสร้างทางด้านโปรแกรม ในบทนี้จะได้กล่าวถึงรายละเอียดต่าง ๆ โดยอ้างอิงการทำงานของเอ็นเบสิก (N-BASIC) อินเตอร์พริเตอร์ของเครื่อง NEC PC 8001

1. แบบของการทำงาน[5] (MODE OF OPERATION)

แบบการทำงานของอินเตอร์พริเตอร์แบ่งออกเป็น 2 แบบดังนี้คือ

ก. แบบไดเรค (DIRECT MODE) เป็นแบบที่รับคำสั่งหนึ่งบรรทัดและปฏิบัติตามคำสั่งนั้นในทันที นอกจากนี้ยังสามารถรับคำสั่งได้หลายคำสั่ง แต่ต้องอยู่ในบรรทัดเดียวกัน (เรียกว่า มัลติสเตทเมนท์ (MULTI-STATEMENT)) ลักษณะที่เห็นได้ชัดในแบบนี้คือ ไม่มีหมายเลขบรรทัดนำหน้า บางคำสั่งถูกบังคับให้สามารถใช้ได้กับแบบไดเรคเท่านั้นเช่น LIST NEW, RUN เป็นต้น

ข. แบบอินไดเรค (INDIRECT MODE) เป็นแบบที่รับคำสั่งที่ต้องการมีหมายเลขบรรทัด

(LINE NUMBER) อยู่ประจำบรรทัดของกลุ่มคำสั่งนั้น อินเตอร์พรีเตอร์จะนำคำสั่งเหล่านี้ไปไว้ในบริเวณหน่วยความจำหลักที่ใช้เก็บโปรแกรม โดยยังไม่มีการปฏิบัติการทำงานจนกว่าจะได้รับคำสั่ง RUN เมื่อได้รับคำสั่ง RUN อินเตอร์พรีเตอร์จะอ่านโปรแกรมในแต่ละบรรทัด มาทำการแปลว่าเป็นคำสั่งใด แล้วจึงทำงานตามคำสั่งนั้น จากนั้นจึงอ่านคำสั่งต่อไปเพื่อทำงานไปเรื่อย ๆ จนหมดโปรแกรม

2. โครงสร้างของข้อมูลในหน่วยความจำ หลัก[1]

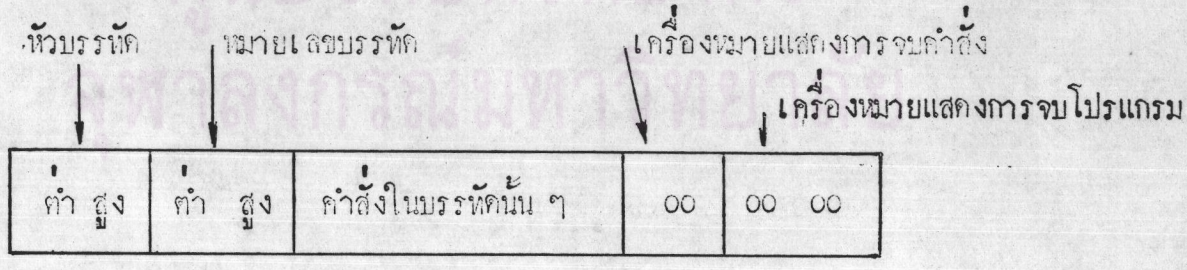
เพื่อเป็นการประหยัดเนื้อที่ในหน่วยความจำ การเก็บข้อมูลหรือโปรแกรมจึงต้องมีการจัดโครงสร้างข้อมูล (DATA STRUCTURE) โดยแบ่งออกดังนี้คือ

ก. คำสั่ง (STATEMENT)

คำสั่งที่ใช้ในภาษาเบสิคเนื่องจากเป็นภาษาชั้นสูง จึงประกอบด้วยตัวอักษรหลายตัวเพื่อให้อ่านแล้วได้ใจความ เช่น PRINT, GOTO, GOSUB เป็นต้น การที่จะนำเอาอักษรเหล่านี้มาเก็บในหน่วยความจำหลักจะเป็นการสิ้นเปลืองเนื้อที่โดยไม่จำเป็น จึงควรแปลงคำสั่งเหล่านี้เป็นรหัส เพื่อลดจำนวนตัวอักษร รหัสนี้เรียกว่า โทเคน (TOKEN)

ข. รูปแบบบรรทัด (LINE FORMAT)

โปรแกรมเมื่ออยู่ในแบบอินไดเรคโปรแกรมจะถูกนำมาเก็บในหน่วยความจำหลัก ซึ่งการเก็บจะมีรูปแบบเพื่อสะดวกในการทำงานของอินเตอร์พรีเตอร์ ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 รูปแบบบรรทัด

การจัดเก็บบรรทัดของโปรแกรม (PROGRAM TEXT) ในหน่วยความจำหลักจะเรียงติดต่อกันทั้งโปรแกรม โดยเรียงลำดับกันตามหมายเลขบรรทัดจากน้อยไปหามาก แต่ละบรรทัดมีรายละเอียดดังนี้

1) หัวบรรทัดแบ่งออกเป็น 2 ไบต์ ซึ่งจะเก็บแอดเดรสของบรรทัดต่อไป (NEXT LINE ADDRESS) โดยไบต์ที่ 1 แทนนัยสำคัญด้านต่ำของแอดเดรส (LOW ORDER BYTE) ไบต์ที่ 2 แทนนัยสำคัญของแอดเดรส (HIGH ORDER BYTE) การจัดเก็บใช้เลขฐานสิบหก

2) หมายเลขบรรทัดแบ่งออกเป็น 2 ไบต์ ซึ่งเก็บหมายเลขบรรทัด โดยไบต์ที่ 1 เป็นไบต์ที่มีนัยสำคัญด้านต่ำของแอดเดรสไบต์ที่ 2 เป็นไบต์ที่มีนัยสำคัญด้านสูงของแอดเดรสการจัดเก็บเป็นเลขฐานสิบหก

3) บรรทัดของโปรแกรม จะจัดเก็บคำสั่งที่ถูกแปลงเป็นโทเคน เรียบร้อยแล้วรวมทั้งอักษรอื่น ๆ ที่เหลือของบรรทัดนั้น

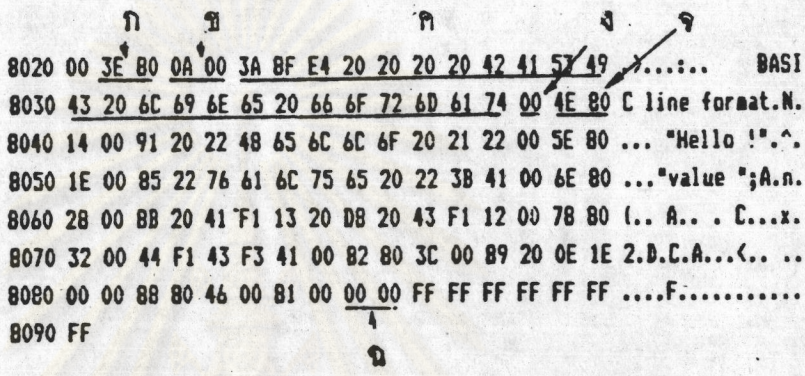
4) เครื่องหมายแสดงการหมดบรรทัด ซึ่งใช้เลขศูนย์จำนวน 1 ไบต์

5) เครื่องหมายแสดงการหมดโปรแกรม ซึ่งใช้เลขศูนย์จำนวน 2 ไบต์ จากรูปที่ 2.2 แสดงตัวอย่างโปรแกรมเบสิคซึ่งเมื่อเก็บในหน่วยความจำหลักจะมีลักษณะดังรูปที่ 2.3

```
10 '    BASIC line format
20 PRINT "Hello !"
30 INPUT"value ";A
40 IF A=2 THEN C=1
50 D=C+A
60 GOTO 30
70 END
```

รูปที่ 2.2 โปรแกรมที่ต้องการเก็บไว้ในหน่วยความจำ

BASIC LINE FORMAT REPRESENTATION IN MEMORY



รูปที่ 2.3 แสดงการเก็บโปรแกรมรูปที่ 2.2 ในหน่วยความจำ

จากรูปที่ 2.3 ส่วนประกอบต่าง ๆ ที่มีหมายเลขกำกับมีความหมายดังนี้

ก) หัวบรรทัดของหมายเลขบรรทัด 10 เก็บแอดเดรสของหมายเลขบรรทัด 20 คือ 803EH

ข) หมายเลขบรรทัดจะเก็บ 00 0AH ซึ่งเป็นเลขฐานสิบหก (เลข 10 ฐานสิบ)

ค) บรรทัดของโปรแกรม คือข้อความต่าง ๆ ภายในหมายเลขบรรทัด 10

ง) เครื่องหมายแสดงการหมดบรรทัด

จ) หัวบรรทัดของหมายเลขบรรทัด 20 ซึ่งเป็นบรรทัดถัดไป

ฉ) เครื่องหมายแสดงการหมดโปรแกรม

3. ค่าคงที่ (CONSTANTS)

ในการเขียนโปรแกรมย่อมมีการกำหนดค่าให้กับตัวแปร ฉะนั้นการเก็บค่าต่าง ๆ จึงมีรูปแบบแตกต่างกันออกไปดังนี้

ก. ค่าคงที่ชนิดสตริง (STRING CONSTANTS) ใช้กำหนดตัวอักษรหรือตัวเลขที่ไม่สามารถนำไปคำนวณให้กับตัวแปรชนิดนี้ แต่ละตัวอักษรหรือตัวเลขนี้ใช้แทนด้วยรหัสแอสกี (ASCII) และใช้เนื้อที่ในหน่วยความจำ จำนวน 1 ไบต์ แทน 1 ตัวอักษร ดังแสดงในรูปที่ 2.4

```
10 N$="STRING Constants representation in memory"
```

```
8020 00 54 80 0A 00 4E 24 F1 22 53 54 52 49 4E 47 20 .T...N$.."STRING
8030 43 6F 6E 73 74 61 6E 74 73 20 72 65 70 72 65 73 Constants repres
8040 65 6E 74 61 74 69 6F 6E 20 69 6E 20 6D 65 6D 6F entation in memo
8050 72 79 22 00 00 00 00 00 00 00 00 00 00 00 00 00 ry".............
8060 00
```

รูป 2.4 แสดงการเก็บค่าคงที่ที่กำหนดให้กับตัวแปร N$ ในหน่วยความจำ

ข. ค่าคงที่ชนิดจำนวนเต็ม (INTEGER CONSTANTS) เป็นเลขจำนวนเต็มที่อยู่ระหว่าง -32767 กับ +32767 การเก็บแบ่งเป็น 3 ลักษณะดังนี้

1) เลขจำนวนเต็มที่มีค่า 0 ถึง 9 จะใช้เนื้อที่ในการเก็บ 1 ไบต์ เครื่องหมายอีก 1 ไบต์ (ถ้ามี) ดังแสดงในรูปที่ 2.5 โดยมีการแทนค่าตัวเลขต่าง ๆ ดังนี้

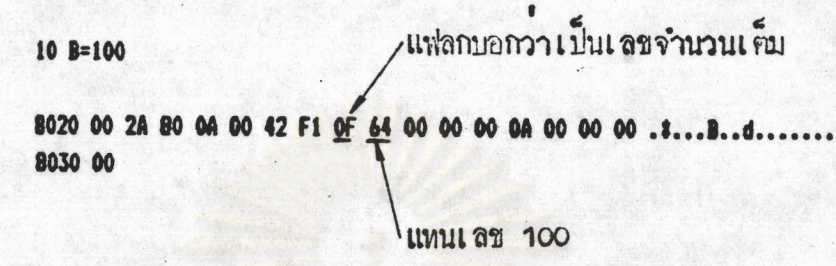
11	แทนเลข	0
12	"	1

13	แทนเลข	2
14	"	3
15	"	4
16	"	5
17	"	6
18	"	7
19	"	8
1A	"	9
F3	แทนเครื่องหมาย	+
F4	"	-

```
10 A=1
                          แทนค่าของตัวเลข 1
8020 00 29 80 0A 00 41 F1 12 00 00 00 00 00 00 00 00 .)...A.........
8030 00
     10 A=-1              แสดงเครื่องหมาย
8020 00 2A 80 0A 00 41 F1 F4 12 00 00 00 0A 00 00 00 .*...A.........
8030 00
                          แทนค่าของตัวเลข 1
```

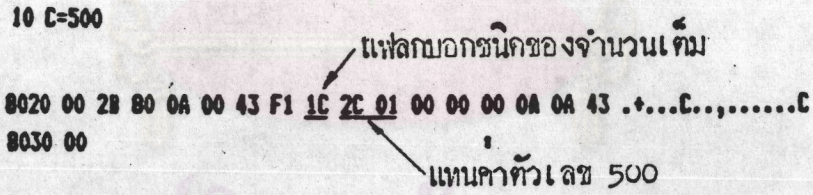
รูปที่ 2.5 แสดงการเก็บค่าคงที่ชนิดจำนวนเต็มแบบที่ 1 ในหน่วยความจำ

2) เลขจำนวนเต็มที่มีค่าตั้งแต่ 10 ถึง 255 จะใช้เนื้อที่ในการเก็บ 1 ไบต์ จัดเก็บเป็นเลขฐานสิบหก โดยมีแฟลก (FLAG) ซึ่งแสดงด้วย OFH จำนวน 1 ไบต์นำหน้า และเครื่องหมาย 1 ไบต์ (ถ้ามี) รวมไม่เกิน 3 ไบต์ ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 แสดงการเก็บค่าคงที่ชนิดจำนวนเต็มแบบที่ 2 ในหน่วยความจำ

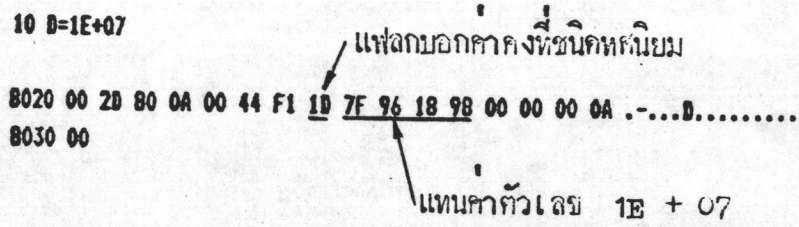
3) เลขจำนวนเต็มที่มีค่าตั้งแต่ 255 ถึง 32767 จะใช้เนื้อที่ในการเก็บค่า 2 ไบต์ จัดเก็บเป็นเลขฐานสิบหก โดยแฟลกซึ่งแสดงด้วย 1CH จำนวน 1 ไบต์ นำหน้าและเครื่องหมาย 1 ไบต์ (ถ้ามี) รวมไม่เกิน 4 ไบต์ แสดงในรูปที่ 2.7



รูปที่ 2.7 แสดงการเก็บค่าคงที่จำนวนเต็มแบบที่ 3 ในหน่วยความจำ

ค. ค่าคงที่ชนิดทศนิยม (SINGLE PRECISION CONSTANTS) เป็นเลขทศนิยมตั้งแต่ 00 ถึง 1E +.38(1×10^{38}) ใช้เนื้อที่ในการเก็บค่า 4 ไบต์ โดยมีแฟลกซึ่งแสดงด้วยจำนวน 1 ไบต์นำหน้าและเครื่องหมายอีก 1 ไบต์ (ถ้ามี) รวมไม่เกิน 6 ไบต์

ค่าที่เก็บจะอยู่ในรูปเลขยกกำลัง โดยภายใน 4 ไบต์ ไบต์สุดท้ายเก็บกำลัง ดังแสดงในรูปที่ 2.8

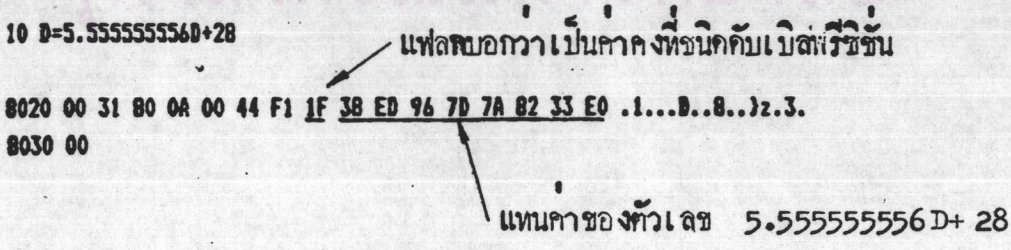


รูปที่ 2.8 แสดงการเก็บค่าคงที่ชนิดทศนิยม

ค่าคงที่ชนิดนี้ โดยปกติถ้าไม่ใส่จุดทศนิยม หรือตัวตาม (SUFFIX) จะใช้เก็บตัวเลขได้ตั้งแต่ 32767 ถึง 999999

ง. ค่าคงที่ชนิดทศนิยมดับเบิลพรีซิชัน (DOUBLE PRECISION CONSTANTS) เป็นเลขทศนิยมตั้งแต่ 0.0 ถึง 1D + 38 (1×10^{38}) แต่เนื่องจากสามารถแสดงจำนวนเต็มหน้าจุดทศนิยมได้ถึง 16 หลัก จึงใช้ D แทน E ซึ่งแสดงได้เพียง 6 หลัก ถ้าต้องการใช้ค่าคงที่ชนิดนี้บางครั้งอาจต้องตามด้วยเครื่องหมาย # หลังตัวคงที่นั้น ๆ เพื่อให้สามารถแสดงเลขหน้าจุดทศนิยมมากกว่า 6 หลัก ค่าคงที่ชนิดใช้เนื้อที่ในการเก็บค่า 8 ไบต์ โดยมีแฟลกซึ่งแสดงด้วย 1FH จำนวน 1 ไบต์ และเครื่องหมายอีก 1 ไบต์ (ถ้ามี) รวมไม่เกิน 10 ไบต์

การจัดเก็บค่ามีลักษณะคล้ายกับค่าคงที่ชนิดทศนิยม เพียงแต่จำนวนไบต์ในการเก็บตัวเลขจำนวนเต็มมากกว่าค่าคงที่ชนิดทศนิยม 4 ไบต์ ฉะนั้นสามารถเก็บค่าได้ละเอียดขึ้นแต่การยกกำลังยังสูงสุดเท่าเดิม คือไม่เกิน 10 +38 ดังแสดงในรูปที่ 2.9



รูป 2.9 แสดงการเก็บค่าคงที่ชนิดทศนิยมดับเบิลพรีซิชั่น

4 ตัวแปรเดี่ยว[1] (SCALAR VARIABLE)

ตัวแปรต่าง ๆ ที่ใช้ในโปรแกรมจะมีที่เก็บชื่อและค่าของตัวแปรอยู่ในส่วนท้ายของโปรแกรม ตัวแปรแต่ละตัวจะใช้เนื้อที่ในหน่วยความจำยาวกี่ไบต์นั้นอยู่กับชนิดของตัวแปรนั้น ๆ ส่วนชื่อของตัวแปรจะยาวเท่าไรก็ได้แต่ตัวอินเตอร์พรีเตอร์จะถือว่ามีนัยสำคัญเพียงสองตัวแรกเท่านั้น ตัวแรกต้องเป็นตัวอักษรเท่านั้นที่เหลืออาจเป็นตัวเลขหรืออักษรก็ได้ นอกจากนั้นการตั้งชื่อต้องไม่ตรงกับคำเฉพาะ (RESERVED WORD) ที่ใช้ในภาษาเบสิค

ชนิดของตัวแปรที่ใช้เก็บค่าแบ่งออกเป็น 4 ชนิดคือ

ก. ตัวแปรสตริง (STRING VARIABLE) ชื่อของตัวแปรชนิดนี้จะมีเครื่องหมายดอลล่าร์ ($) ตามหลังชื่อตัวแปรเสมอ เช่น AN$, CI$, ABC$ ตัวแปรนี้ใช้เก็บเลขหรือตัวอักษรก็ได้ การกำหนดค่าให้ตัวแปรชนิดนี้จะใช้ค่าคงที่ชนิดสตริง

การเก็บภายในหน่วยความจำนั้น ตัวแปรถูกเก็บไว้ท้ายโปรแกรมจะใช้เนื้อที่ 6 ไบต์ต่อตัวแปรหนึ่ง ๆ ดังนี้ ดังแสดงในรูปที่ 2.10

1) ไบต์แรกจะเป็นไบต์บอกชนิดของตัวแปรแทนด้วยรหัส 03

2) ไบต์ต่อมาเก็บชื่อของตัวแปรตัวที่2และอีก 1 ไบต์เก็บชื่อตัวแรกของตัวแปรการเก็บชื่อตัวที่ 2 ของตัวแปรก่อน เพื่อเพิ่มประสิทธิภาพและความเร็วในการค้นหา เพราะว่าส่วนใหญ่เวลาตั้งชื่อตัวแป จะเป็นดังนี้ A1$, A2$ เป็นต้น

3) ไบต์ที่ 4 ใช้เก็บความยาวของตัวแปรที่เก็บภายในสตริงแสตค (STRING STACK) ซึ่งเป็นส่วนของหน่วยความจำที่ใช้เก็บตัวอักษรหรือตัวเลขของตัวแปรชนิดนี้

4) ไบต์ที่ 5 และ 6 ใช้เก็บแอดเดรสที่ชี้ไปยังสตริงแสตค ตัวอักษรที่เก็บในสตริงสแตคนั้นจะเก็บเป็นรหัสแอสกี

```
10 B$="Test"
20 INPUT"STRING TEST ";G$
30 END
```

```
8020 00 2F 80 0A 00 42 24 F1 22 54 65 73 74 22 00 45 ./...B$."Test".E
8030 80 14 00 85 22 53 54 52 49 4E 47 20 54 45 53 54 ...."STRING TEST
8040 22 3B 47 24 00 4B 80 1E 00 81 00 00 00 03 00 42 ";G$.K.........B
8050 04 29 80 03 00 47 17 E9 E9 00 00 00 00 00 00 00 .)...G.........
          1  2  3  4

                STRING STACK

E9E0 00 00 00 00 00 00 00 00 00 48 65 6C 6C 6F 20 53 .........Hello S
E9F0 54 52 49 4E 47 20 53 54 41 43 4B 20 74 65 73 74 TRING STACK test
```

รูปที่ 2.10 แสดงการเก็บตัวแปรชนิดสตริง ในหน่วยความจำ

ข. ตัวแปรชนิดจำนวนเต็ม (INTEGER VARIABLE) ตัวแปรชนิดนี้ใช้กับเลขจำนวนเต็มที่อยู่ระหว่าง -32767 ถึง 32767 ชื่อของตัวแปรที่เป็นเลขจำนวนเต็มจะมีเครื่องหมายเปอร์เซ็นต์ (%) ตามหลังชื่อตัวแปร เช่น AB%, MA%, C2% เป็นต้น รายละเอียดการเก็บเป็นดังนี้ดังแสดงในรูปที่ 2.11

1) ไบต์แรกจะเป็นไบต์ที่บอกชนิดของตัวแปรแทนด้วยรหัส 02

2) ไบต์ต่อมาเก็บชื่อตัวที่ 2 ของตัวแปร และอีก 1 ไบต์จะเก็บชื่อตัวแรกของตัวแปร

3) ไบต์ที่ 4 และ 5 ใช้เก็บค่าของตัวแปรโดยเก็บเป็นเลขฐานสิบหก

```
10 AB%=-2
20 CD%=200
30 EF%=5000

8020 00 2C 80 0A 00 41 42 25 F1 F4 13 00 37 80 14 00 .,...AB%....7...
8030 43 44 25 F1 0F C8 00 43 80 1E 00 45 46 25 F1 1C CD%....C...EF%..
8040 88 13 00 00 00 02 42 41 FE FF 02 44 43 C8 00 02 ......BA...DC...
8050 46 45 88 13 02 00 00 00 00 C0 00 00 00 00 00 00 FE..............
8060 00
                      1     2     3
```

รูปที่ 2.11 แสดงการเก็บตัวแปรชนิดจำนวนเต็มในหน่วยความจำ

ค. ตัวแปรชนิดเลขทศนิยม (SINGLE PRECISION VARIABLES) ตัวแปรชนิดนี้ใช้เก็บตัวเลขที่มีจำนวนระหว่าง 1.0E -38 ถึง 1.0E + 38 ตัวแปรชนิดนี้ไม่จำเป็นต้องมีเครื่องหมายตามหลังชื่อแต่ถ้าจะมีก็ใช้เครื่องหมายอัศเจรีย์ (!) เช่น AB, CD!, M! เป็นต้น รายละเอียดการเก็บต่าง ๆ ในหน่วยความจำเป็นดังนี้ ดังแสดงในรูปที่ 2.12

1) ไบต์แรกจะเป็นไบต์ที่บอกชนิดของตัวแปรแทนด้วยรหัส 04

2) ไบต์ ถัดมาจะเก็บชื่อตัวที่ 2 ของตัวแปรและอีก 1 ไบต์ จะเก็บชื่อตัวแรกของตัวแปร

3) ไบต์ที่ 4 ถึง 7 ใช้เก็บค่าของตัวแปร โดยไบต์ที่ 7 ใช้เก็บเลขยกกำลังสองไบต์ที่ 6, 5, และ 4 ใช้เก็บค่าเป็นเลขฐานสิบหก ซึ่งมีนัยสำคัญของแต่ละไบต์ตามลำดับ

```
10 AB!=-2
20 CD!=200
30 EF!=5000

8020 00 2C 80 0A 00 41 42 21 F1 F4 13 00 37 80 14 00 .,...AB!....7...
8030 43 44 21 F1 0F C8 00 43 80 1E 00 45 46 21 F1 1C CD!....C...EF!..
8040 88 13 00 00 00 04 42 41 00 00 80 82 04 44 43 00 ......BA.....DC.
8050 00 48 88 04 46 45 00 40 1C 8D 1E 00 00 00 00 00 .H..FE.@........
8060 00
                      1     2     3
```

รูปที่ 2.12 แสดงการเก็บตัวแปรชนิดตัวเลขทศนิยมในหน่วยความจำ

ง. ตัวแปรชนิดเลขทศนิยมดับเบิลพรีซิชั่น (DOUBLE PRECISION VARIABLES)
ตัวแปรชนิดนี้ใช้เก็บตัวเลขที่มีจำนวนเท่ากับ ตัวแปรทศนิยมแต่สามารถเก็บตัวเลขได้ละเอียดกว่าตัวแปรชนิดทศนิยม ตัวแปรที่แสดงว่าเป็นตัวแปรชนิดนี้ จะมีเครื่องหมาย# ต่อท้ายชื่อตัวแปร เช่น AB#, CD#, M# เป็นต้น

การเก็บภายในหน่วยความจำนั้น ตัวแปรจะถูกเก็บไว้ท้ายโปรแกรม จะใช้เนื้อที่ 11 ไบต์ ต่อตัวแปรหนึ่ง ๆ ดังนี้ ดังแสดงในรูปที่ 2.13

1) ไบต์แรกจะเป็นไบต์ที่บอกชนิดของตัวแปรแทนด้วยรหัส 08

2) ไบต์ต่อมาจะเก็บชื่อตัวที่ 2 ของตัวแปรและอีก 1 ไบต์จะเก็บชื่อตัวแรกของตัวแปร

3) ไบต์ 4 ถึง 11 ใช้เก็บค่าตัวแปร โดยไบต์ที่ 11 ใช้เก็บเลขยกกำลังของสองไบต์ 4 ถึง 10 ใช้เก็บค่าเป็นเลขฐานสิบหก ซึ่งมีนัยสำคัญของแต่ละไบต์ตามลำดับ

```
10 AB#=-2
20 CD#=200
30 EF#=5000

8020 00 2C B0 0A 00 41 42 23 F1 F4 13 00 37 B0 14 00 .,...AB#....7...
8030 43 44 23 F1 0F C8 00 43 B0 1E 00 45 46 23 F1 1C CD#....C...EF#..
8040 B8 13 00 00 00 08 42 41 00 00 00 00 00 00 B0 B2 ......BA........
8050 08 44 43 00 00 00 00 00 00 48 B8 0B 46 45 00 00 .DC......H..FE..
8060 00 00 00 40 1C BD 08 00 00 00 00 00 00 00 00 00 ...@............
8070 00
```

1 2 3

รูปที่ 2.13 แสดงการเก็บตัวแปรชนิดเลขทศนิยมดับเบิลพรีซิชั่นในหน่วยความจำ

5 ตัวแปรชุด[1] (ARRAY VARIABLE)

ตัวแปรชุดสามารถสร้างได้โดยคำสั่ง DIM และสามารถกำหนดจำนวนมิติได้มากที่สุด 255 มิติ ซึ่งแต่ละมิติมีค่าได้ระหว่าง 0 ถึง 65536

ถ้ามีการใช้ตัวแปรภายในโปรแกรมโดยไม่ใช้คำสั่ง DIM ไว้ล่วงหน้า กรณีนี้สามารถใช้ได้เพียงมิติเดียว และมีค่าระหว่าง 0 ถึง 10 (ฐาน 16)

การเก็บตัวแปรเหล่านี้ในหน่วยความจำหลักมีรูปแบบดังนี้ ดังแสดงในรูปที่ 2.14

1. ไบต์แรกใช้เก็บชนิดของตัวแปรนั้น ๆ
2. ไบต์ต่อมาใช้เก็บชื่อตัวที่ 2 ของตัวแปร และอีก 1 ไบต์ ใช้เก็บเป็นชื่อตัวแรกของตัวแปรนั้น
3. ไบต์ที่ 4 และ 5 ใช้เก็บความยาวของตัวแปรหมวดนี้งคำนวณได้ดังนี้

ความยาว = จำนวนไบต์ในการเก็บค่าของตัวแปรชนิดนั้น ๆ + จำนวนมิติ × 2 + 1

4. ไบต์ที่ 6 ใช้เก็บจำนวนมิติ
5. ไบต์ต่อ ๆ ไปใช้เก็บค่าสูงสุดของมิติ ๆ ละ 2 ไบต์
6. ไบต์ต่อจากไบต์ที่ใช้เก็บค่าของมิติจะใช้เก็บค่าของตัวแปรหมวดจำนวนไบต์ขึ้นกับตัวแปร

```
10 DIM D$(1,2,3)
20 D$(1,1,2)="String array"

8020 00 31 80 0A 00 86 20 44 24 28 12 2C 13 2C 14 29 .1.... D$(.,.,.)
8030 00 4E 80 14 00 44 24 28 12 2C 12 2C 13 29 F1 22 .N...D$(.,.,.)."
8040 53 74 72 69 6E 67 20 61 72 72 61 79 22 00 00 00 String array"...
8050 03 00 44 4F 00 03 04 00 03 00 02 00 00 00 00 00 ..DO............
8060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8080 00 00 00 00 00 00 00 00 00 0C 4C 80 00 00 00 00 ..........L.....
8090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
80A0 00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF ................
80B0 FF
```

รูปที่ 2.14 แสดงการเก็บตัวแปรหมวดในหน่วยหน่วยความจำ

6. การแบ่งหน่วยความจำ[1,5] (Memory Map)

การจัดการในหน่วยความจำเพื่อแบ่งหน่วยความจำในการใช้งาน สามารถแบ่งออกเป็นส่วน ๆ ดังนี้ ดังแสดงในรูป 2.15

ก. หน่วยความจำที่เก็บอินเตอร์พรีเตอร์ภาษาเบสิกซึ่งปกติจะเป็นหน่วยความจำถาวร (READ ONLY MEMORY หรือ ROM)

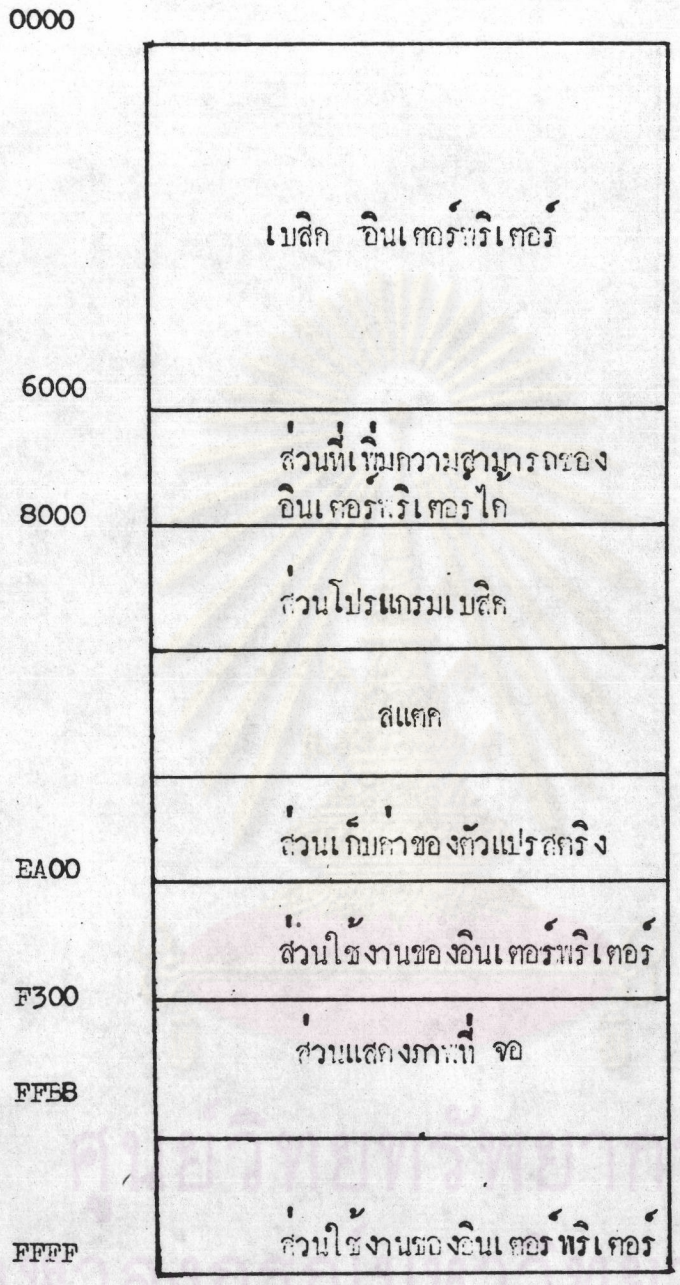
ข. หน่วยความจำส่วนที่เก็บโปรแกรมภาษาเบสิก และตัวแปรต่าง ๆ

ค. หน่วยความจำที่ทำหน้าที่เป็นสเตค

ง. หน่วยความจำส่วนที่เก็บค่าคงที่ชนิดสตริง

จ. หน่วยความจำที่ใช้งานทั่วไป

ฉ. หน่วยความจำส่วนที่เก็บข้อมูลสำหรับแสดงผลบนจอภาพ



รูปที่ 2.15 แสดงการแบ่งหน่วยความจำ

7. โครงสร้างทางด้านโปรแกรมของอินเตอร์พริเตอร์ [6]

การทำงานโดยทั่วไปของอินเตอร์พริเตอร์จะเป็นดังต่อไปนี้ เมื่ออินเตอร์พริเตอร์รับคำสั่งมาจากแป้นพิมพ์คำสั่งเหล่านั้นจะถูกเก็บไว้ในบัฟเฟอร์ (BUFFER) แล้วจะมีโปรแกรมที่เรียกว่า เล็กซิเคิล แอนนาไลซ์เซอร์ (LEXICAL ANALYSER) และโปรแกรม เท็คอะตอมไมเซอร์ (TEXT ATOMIZER) มาทำการแปลงและเก็บชุดของคำสั่งเหล่านั้นในรูปของโทเคน โดยโปรแกรมเล็กซิเคิลแอนนาไลซ์เซอร์ จะทำหน้าที่ในการวิเคราะห์กลุ่มของตัวอักษรที่ถูกใส่มาทางแป้นพิมพ์ว่าส่วนไหนคือคำสั่งและส่วนไหนไม่ใช่คำสั่ง แล้วส่งผลการวิเคราะห์ไปโปรแกรมเท็คอะตอมไมเซอร์ ทำหน้าที่แปลงชุดคำสั่งเหล่านั้นให้อยู่ในรูปของโทเคน แล้วนำไปเก็บในหน่วยความจำในส่วนที่ใช้เก็บโปรแกรมเพื่อรอการปฏิบัติการตามคำสั่งต่อไป เมื่ออินเตอร์พริเตอร์ได้รับคำสั่ง RUN ชุดของคำสั่งที่ถูกแปลงเก็บเอาไว้ในหน่วยความจำก็จะถูกนำมาวิเคราะห์ทีละคำสั่งว่าเป็นคำสั่งใด แล้วอินเตอร์พริเตอร์ก็จะทำการสั่งการควบคุมไปให้โปรแกรมย่อยที่ทำหน้าที่ตามคำสั่งนั้น ๆ เพื่อตรวจสอบความถูกต้องของคำสั่งและปฏิบัติการตามเป้าหมายของคำสั่งนั้นจนกว่าจะสำเร็จ แล้วจึงส่งการควบคุมมายังอินเตอร์พริเตอร์เพื่ออ่านคำสั่งต่อไปเช่นนี้เรื่อย ๆ จนกว่าจะหมดโปรแกรม

จากการทำงานของอินเตอร์พริเตอร์ดังได้กล่าวไปแล้ว จะเห็นว่าอินเตอร์พริเตอร์ประกอบด้วยโปรแกรมต่าง ๆ ดังนี้คือ

ก. โปรแกรมเล็กซิเคิล แอนนาไลซเซอร์ (LEXICAL ANALYSER) ทำหน้าที่ในการวิเคราะห์กลุ่มของตัวอักษรที่ถูกใส่เข้ามาทางแป้นพิมพ์ว่ากลุ่มของตัวอักษรใดเป็นคำสั่ง กลุ่มใดที่ไม่ใช่คำสั่ง เช่นถ้าเจอกลุ่มตัวอักษร PRINT ก็รู้ว่ากลุ่มของตัวอักษรนี้เป็นคำสั่ง แต่ถ้าเจอกลุ่มตัวอักษร PRINT ก็รู้ว่ากลุ่มอักษรนี้ไม่ใช่คำสั่ง การตรวจสอบเหล่านี้ได้อาศัยหลักการของการพาร์ซิ่ง (PARSING)

ข. โปรแกรมเท็คอะตอมไมเซอร์ (TEXT ATOMIZER) ทำหน้าที่ในการแปลงกลุ่มของตัวอักษรที่ถูกใส่เข้ามาทางแป้นพิมพ์เป็นรหัสที่สั้นกว่าเรียกว่าโทเคน เพื่อเป็นการประหยัดเนื้อที่ในหน่วยความจำและทำหน้าที่ในการจัดรูปแบบของคำสั่งแต่ละบรรทัดและโครงของข้อมูลภายในให้เป็นไปตามรูปแบบมาตรฐานตามที่กล่าวไปแล้วข้างต้น

ค. ส่วนที่ทำหน้าที่ควบคุมการทำงานของโปรแกรม ทำหน้าที่คอยวิเคราะห์ว่าคำสั่งที่ได้รับ

มาเป็นคำสั่งอะไร แล้วส่งการควบคุมไปยังโปรแกรมย่อยที่ทำหน้าที่ตามคำสั่งนั้น ๆ

ง ส่วนที่ทำหน้าที่ประมวลผลและวิเคราะห์ความถูกต้องของขั้นตอนและรูปแบบของคำสั่ง (PROCESS AND SYNTAX ANALYSER) เป็นส่วนที่ทำหน้าที่ประมวลผลให้เป็นไปตามจุดมุ่งหมายของแต่ละคำสั่ง และทำการวิเคราะห์ความถูกต้องของรูปแบบของคำสั่งร่วมกันไปด้วย

8. ตารางต่าง ๆ ในอินเตอร์พรีเตอร์ [6]

อินเตอร์พรีเตอร์ของเอ็นเบสิคตัวนี้มีตารางต่าง ๆ ที่สำคัญสี่ตาราง โดย 3 ตารางแรกคือ ตารางของคำสั่ง, ตารางเวียร์อะคอมไมด์เซชั่น, ตารางบอกความผิดพลาด จะเรียงติดต่อกันโดยเริ่มตั้งแต่ตำแหน่ง $(33BD)_{16}$ **ส่วนตารางเก็บค่าตัวแปรอยู่ส่วนท้ายของโปรแกรม**

ก. ตารางของคำสั่ง (COMMAND TABLE) เป็นตารางเก็บแอดเดรสซึ่งเป็นจุดเริ่มต้นของโปรแกรมย่อยในอินเตอร์พรีเตอร์ที่ใช้ตีความและปฏิบัติตามคำสั่งแต่ละแบบโดยใช้เนื้อที่ในการเก็บ 2 ไบต์ และเรียงคำสั่งต่าง ๆ ตามค่าของโทเคน (TOKEN) ที่ชี้แทนคำสั่งนั้นโดยเรียงจากน้อยไปมาก ดังรูปที่ 2.16

```
TABLE OF COMMAND STATEMENT

        DW      4S2FH           ; END
        DW      4159H           ; FOR
        DW      4AGSH           ; NEXT
        DW      45BEH           ;DATA
        DW      4E37H           ; DIM
        DW      4939H           ; READ
```

รูปที่ 2.16 แสดงตารางของคำสั่งต่าง ๆ

ช. ตารางเวิร์ดอตอมไมค์เซชั่น (VERB ATOMIZATION TABLE)

ใช้ในการเปรียบเทียบเวิร์ดเนม (คือคำสั่งต่าง ๆ ที่อยู่ในรูปภาษาอังกฤษ เช่น PRINT, LIST, RUN ฯลฯ) ว่าเป็นเวิร์ดเนมที่ถูกต้องหรือไม่ และถ้าเป็นเวิร์ดเนมที่ถูกต้องจะแทนด้วยค่าของโทเคนเท่าไร ตารางนี้แบ่งออกเป็นสองส่วนคือ

1) ส่วนที่เก็บแอดเดรสจุดเริ่มต้นของคีย์เวิร์ด (KEY WORD) ต่าง ๆ โดยเรียงลำดับ A ถึง Z ใช้เนื้อที่ 2 ไบต์ 1 คีย์เวิร์ด

2) เก็บรายละเอียดของเวิร์ดเนม (VERB NAME) โดยเวิร์ดเนมแต่ละตัวจะตามด้วยโทเคนของเวิร์ดเนมนั้น ๆ และเมื่อหมดเวิร์ดเนมของแต่ละคีย์เวิร์ดจะลงท้ายด้วย 00 (ZERO) จำนวน 1 ไบต์ จะเป็นแบบนี้ไปจนกระทั่งคีย์เวิร์ดสุดท้าย ดังรูปที่ 2.17

```
                KEY   WORD   TABLE

                DW      6138H           ;TOPA
                DW      614EH           ;TOPB
                DW      6157H           ;TOPC


                DW      6402H           ;TOPZ

6138H TOPA:     DB      'N', 'D' + 80H, 0F8H
                DB      'B', 'S', + 80H, 6
                DB      'T', 'N' + 80H, 0EH
                DB      00
        TOPB:
```

รูปที่ 2.17 แสดงตารางเวิร์ดอตอมไมค์เซชั่น

ค. <u>ตารางบอกความผิดพลาด</u> (ERROR MESSAGE TABLE) ใช้เก็บข้อความต่าง ๆ เพื่อแสดงออกมาเมื่อมีความผิดพลาดต่าง ๆ เกิดขึ้น ประกอบด้วยส่วนต่าง ๆ ดังนี้

1) เลข 0 (ZERO) นำหน้าตาราง จำนวน 1 ไบต์

2) ข้อความแสดงการผิดพลาดที่ต้องการให้แสดงออกมาทางจอภาพ เมื่อเกิดความผิดพลาดต่าง ๆ เกิดขึ้น ข้อความจะถูกปิดท้ายด้วย 0 (ZERO) จำนวน 1 ไบต์

```
BASIC   ERROR   MESSAGES   TABLE

        DB      0
        DB      'NEXT WITHOUT FOR',0
        DB      'SYNTAX ERROR', 0
        DB      'RETURN WITHOUT GOSUB', 0
        DB      'OUT OF DATA', 0
```

รูปที่ 2.18 แสดงตารางบอกความผิดพลาด

ง. <u>ตารางเก็บค่าของตัวแปร</u> (VARIABLE TABLE) ใช้ในการเก็บตัวแปรต่าง ๆ ในโปรแกรมซึ่งจะแบ่งเป็น 2 ส่วน คือ

1) <u>ตัวแปรเดี่ยว</u> (SCALAR VARIABLE) จะเก็บตัวแปรที่เป็นตัวแปรตัวเดี่ยว โดด ๆ อาจเป็นตัวแปรทศนิยม ตัวแปรจำนวนเต็มหรืออื่น ๆ มีรูปแบบการเก็บดังได้กล่าวแล้ว โดยตัวแปรแต่ละตัวจะเรียงต่อ ๆ ไปจนหมดทุกตัวแปร ตัวแปรเหล่านี้ถูกเก็บส่วนแรกของตาราง

2) <u>ตัวแปรชุด</u> (ARRAY VARIABLE) ได้แก่ตัวแปรที่เกิดจากคำสั่ง DIM ซึ่งตัวแปรเหล่านี้จะถูกเก็บไว้ต่อจากส่วนที่เป็นตัวแปรเดี่ยว

9. <u>สรุป</u>

จะเห็นว่าแบบการทำงานของอินเตอร์พรีเตอร์มี 2 แบบ คือ

ก. แบบไดเรค (DIRECT MODE)

ข. แบบอินไดเรค (INDIRECT MODE)

ส่วนประกอบของอินเตอร์พรีเตอร์ ประกอบด้วย 2 ส่วนใหญ่ ๆ ด้วยกัน คือ

ก. ตัวโปรแกรมของอินเตอร์พรีเตอร์ แบ่งได้เป็น 2 ส่วน คือ

1) โครงสร้างของโปรแกรม แบ่งหน้าที่การทำงานของโปรแกรม

2) โครงสร้างข้อมูล บอกลักษณะการเก็บข้อมูลชนิดต่าง ๆ

ข. ตารางต่าง ๆ ของอินเตอร์พรีเตอร์

การที่ได้ทราบส่วนประกอบ รายละเอียดการใช้, การปฏิบัติการและรูปแบบต่าง ๆ ของอินเตอร์พรีเตอร์ จะเป็นพื้นฐานในการที่จะทำการดัดแปลงอินเตอร์พรีเตอร์ให้มีความสามารถเพิ่มขึ้น ซึ่งการดัดแปลงแก้ไขอินเตอร์พรีเตอร์จะได้กล่าวในบทที่ 3