บทที่ 3

พื้นฐานภาษาแอสแซมบลี

ASSEMBLY LANGUAGE REQUIREMENTS

วัตถุประสงค์

เมื่อท่านศึกษาบทนี้แล้วท่านจะเข้าใจดังต่อไปนี้

- การกำหนดฟิลด์ต่างๆของภาษาแอสแซมบลี
- การใช้คำสั่งเทียม
- โครงสร้างของโปรแกรมภาษาแอสแซมบลี
- การเขียนรหัสคำสั่งภาษาแอสแซมบลี

CT 215 47

บทที่ 3
พื้นฐานภาษาแอสแซมบลี
ASSEMBLY LANGUAGE REQUIREMENTS

บทนำ

บทนี้เป็นการอธิบายพื้นฐานของรูปแบบการเขียนโปรแกรมภาษาแอสแซมบลี รวมทั้งการใช้ COMMENT รูปแบบโดยๆทั่วๆไปของการเขียนโปรแกรมภาษาแอสแซมบลี คำสั่งเทียม (DIRECTIVES) เป็นองค์ประกอบอันหนึ่งที่ใช้ในการเขียนโปรแกรม คำสั่งเทียมใช้สำหรับการกำหนดเซคเม้นต์ และ PROCEDURE ดูจากตัวอย่างจะมีการอธิบายอยู่ในรูปแบบโครงสร้างของภาษาแอสแซมบลี้ เริ่มแรกของการเขียนโปรแกรมภาษาแอสแซมบลี และจนกระทั่งถึงจุดสุดท้ายของโปรแกรมที่จะทำการเอ็กซีคิวส์

รูปแบบภาษาแอสแซมบลี ASSEMBLY LANGUAGE COMMENTS

การเขียนคำอธิบายการทำงานของคำสั่ง (COMMENT) ในการเขียนโปรแกรมภาษาแอสแซมบลีจะต้องอธิบายให้ชัดเจน ตามวัตถุประสงค์ของชุดคำสั่ง การเขียน COMMENT จะต้องเริ่มต้นด้วย SEMICOLON (;) ถ้าแอสแซมเบลอร์ตรวจพบเซมิโคลอนมันจะถือว่าข้อมความที่ตามมาข้างหลัง เป็นคำอธิบายการทำงานของโปรแกรม COMMENT อาจเขียนเต็มบรรทัดก็ได้หรือตามหลังรหัสคำสั่งก็ได้ ดูจากตัวอย่างต่อไปนี้

```
1.    ;THIS ENTIRE  LINE IS A COMMENT
2.    ADD    AX,BX        ;COMMENT O SAME LINE AS INSTRUCTION
```

COMMENT ที่ตามหลังเซมิโคลอนทั้งหมด แอสแซมเบลอจะไม่เปลี่ยนข้อความเหล่านี้เป็นภาษาเครื่อง ไม่มีผลต่อการทำงานของโปรแกรมหนังสือเล่มนี้ คำสั่งแอสแซมบลีใช้ตัวอักษรตัวใหญ่ ตัวเลข ส่วน COMMENT ใช้ตัวอักษรตัวเล็ก เพื่อสะดวกในการศึกษาโปรแกรม

รูปแบบรหัส CODING FORMAT

โดยทั่วๆไป NAME จะอ้างถึงแอดเดรสของข้อมูล ขณะที่ LABEL จะอ้างถึงแอดเดรสของคำสั่ง แต่กฎเกณฑ์ในการประยุกต์ใช้งานเหมือนกัน ทั้ง NAME และ LABEL บทนี้จะใช้คำว่า NAME ใช้แทนได้ทั้ง NAME และ LABEL รูปแบบทั่วๆไปของคำสั่งมีดังนี้

[NAME] OPERATION OPERAND(S)

คำว่า NAME , OPERATION , และ OPERAND จะถูกแยกส่วนการใช้ช่องว่าง 1 ตัวอักษรหรือ 1 TAP สามารถกำหนดตัวอักขระได้สูงสุด 132 ต่อบรรทัด แต่ส่วนมากจะใช้ 80 ตัวอักขระ เพราะจอภาพมีเพียง 80 ตัวอักขระเท่านั้น

NAME	OPERATION	OPERAND	COMMENT
COUNT	DB	1	;NAME OPERATION , ONE OPERAND
	MOV	AX,0	;OPERATION ,TWO OPERANDS

NAME ,OPERATION และ OPERAND อาจจะเริ่มต้นได้ทุกคอลัมน์ อย่างไรก็ตามเราจะเริ่มต้นที่คอลัมน์เดียวกัน สำหรับการเขียนโปรแกรม โปรแกรมส่วนมากจะใช้ EDITOR ช่วยในการเขียนโปรแกรม

NAME

NAME หรือ LABEL สามารถใช้ตัวอักษรดังต่อไปนี้

ตัวอักษร	A - Z และ a - z
ตัวเลข	0 - 9
เครื่องหมายพิเศษ	Question mark (?)
	Period (.) เฉพาะตัวแรก
	At @
	Underline (_)
	Dollar ($)

ตัวอักขระตัวแรกของ NAME จะต้องนำหน้าด้วยตัวอักษร หรือเครื่องหมายพิเศษ แอสแซมเบลอร์สามารถเข้าใจทั้งตัวเล็กและตัวใหญ่ ความยาวไม่เกิน 31 ตัวอักษร ตัวอย่าง NAME คือ COUNT, PAGE25 , $E10 ส่วนชื่อของรีจีสเตอร์ AX ,DI และ AL เป็นคำสงวนที่อ้างถึงรีจีสเตอร์ใช้เป็น NAME ไม่ได้

```
ADD         AX,BX
```

จากคำสั่งแอสแซมเบลอร์จะรู้ว่า AX และ BX เป็นชื่อที่อ้างอิงของรีจีสเตอร์ ถ้าคำสั่งเป็น

```
MOV        REGSAVE,AX
```

แอสแซมเบลอก็สามารถจำชื่อ REGSAVE ถ้ากำหนดไว้ในโปรแกรม

OPERATION

การกำหนดรายการข้อมูลในโปรแกรมภาษาแอสแซมบลี เช่น DB หรือ DW เป็นการกำหนดฟิลด์ของข้อมูล หรือค่าคงที่ เช่นคำสั่งแสดงการทำงานคือ MOV หรือ ADD เป็นส่วนของ OPERATION

OPERAND

รายการข้อมูล หรือตัวประกอบการทำงานของคำสั่ง ที่กำหนดค่าดริ่มแรกตามข้อกำหนดของรายการข้อมูลภายใต้ชื่อ COUNTER รายการข้อมูลหรือเรียกอีกอย่างหนึ่งว่า โอเปอร์รานด์ จะเป็นตัวกำหนดค่าดังนี้

```
NAME        OPERATION       OPERAND         COMMENT
COUNTER     DB              0               ;DEFINE BYTE (DB) WITH 0
```

สำหรับการทำงานของคำสั่ง โอเปอร์รานด์จะเป็นแสดงการทำงาน โอเปอร์รานด์อาจจะมี 1 ตัว หรือ 2 ตัว หรือไม่มีก็ได้ ดังตัวอย่าง

```
                    OPERATION       OPERAND     COMMENT

NO OPERAND          RET                         ;RETURN
ONE OPERAND         INC             CX          ;INCREMENT CX
TWO OPERANDS        ADD             AX,12       ;ADD 12 TO AX
```

คำสั่งเทียม DIRECTIVES

คำสั่งเทียมจะช่วยสนับสนุนการเขียนโปรแกรมภาษาแอสแซมบลี้ให้สะดวกยิ่งขึ้น คำสั่งเทียมนี้เมื่อเขียนรวมกับคำสั่งภาษาแอสแซมบลี้จะไม่ถูกแปลเป็นภาษาเครื่อง หรือบางครั้งคำสั่งเทียมนี้เรียกว่า PSEUDO-CODE

LISTING DIRECTIVES : PAGE AND TITLE

คำสั่ง PAGE และ TITLE จะช่วยควบคุมรูปแบบของรายการพิมพ์ภาษาแอสแซมบลี

PAGE เป็นคำสั่งเทียมที่บอกจุดเริ่มต้นของการเขียนโปรแกรม ที่บอกจำนวนบรรทัดสูงสุดในแต่ละหน้า และจำนวนตัวอักขระสูงสุดในแต่ละบรรทัด มีรูปแบบการใช้ดังนี้

```
PAGE  [LENGTH],[WIDTH]
```

จากตัวอย่างการใช้คำสั่ง PAGE ซึ่งกำหนด 60 บรรทัดต่อหน้าและ 132 อักขระต่อบรรทัด

```
PAGE   60,132
```

จำนวนบรรทัดต่อหน้าสามารถกำหนดได้ 10 ถึง 255 บรรทัด และตัวอักษรต่อบรรทัดกำหนดได้ถึง 60 ถึง 132 ตัวอักขระ ถ้าเป็น PAGE อย่างเดียวแอสแซมเบลอร์จะกำหนด 50 บรรทัด บรรทัดละ 80 ตัว

TITLE ท่านสามารถใช้คำสั่งเทียม TITLE บอกหัวเรื่องของโปรแกรมที่พิมพ์ 2 บรรทัดในแต่ละหน้า มีรูปแบบการเขียนดังนี้

```
TITLE   text
```

ถ้าโปรแกรมมีชื่อ ASMSORT จะเป็นชื่อย่อจะต้องมีคำอธิบายสามารถมีความยาวได้ถึง 60 ตัวดังนี้

```
TITLE ASMSORT Assembly program to sort customer name
```

SEGMENT DIRECTIVES

โปรแกรมภาษาแอสแซมบลี้ประกอบด้วยหนึ่งเซคเม้นต์หรือมากกว่าหนึ่งเซคเม้นต์ ในส่วนของ CODE SEGMENT จะใช้สำหรับการเอ็กซีคิวส์รหัสคำสั่ง ส่วน DATA SEGMENT จะใช้ในการกำหนดรายการข้อมูล และ STACK SEGMENT กำหนดพื้นที่ของสแตคในหน่วยความจำ คำสั่งเทียมที่ใช้ในการกำหนดเซคเม้นต์ คือ คำสั่ง **SEGMENT** มีรูปแบบการเขียนดังนี้

NAME	OPERATION	OPERAND	COMMENT
NAME	SEGMENT	[OPTION]	;BEGIN SEGMENT
-	-		
NAME	ENDS		;END SEGMENT

ชื่อของเซคเม้นต์จะต้องอยู่บนสุด และการตั้งชื่อควรจะใช่ชื่อง่ายๆ ส่วนคำสั่ง ENDS เป็นตัวกำหนดจุดสุดท้ายของเซคเม้นต์ และต้องมีชื่อกำหนดเหมือนกับชื่อเริ่มต้นของเซคเม้นต์ ขนาดของเซคเม้นต์จะมีหน่วยความจำ 64 K คำสั่งเทียมเซคเม้นต์ (SEGMENT) จะมี OPTION อยู่ 3 ชนิดคือ ALIGNMENT ,COMBINE และ CLASS มีรูปแบบการเขียนดังนี้

NAME	SEGMENT	ALIGN COMBINE 'CLASS'

ALIGNMENT TYPE วิธีการกำหนดแบบนี้จะกำหนดขอบเขตในเซคเม้นต์ที่เริ่มต้น สำหรับชนิดที่ต้องการกำหนดขอบเขตใช้คำสั่ง PARA ซึ่งกำหนดขอบเขตของพารากราฟ (PARAGRAPH) ดังนั้นแอดเดรสเริ่มแรกจะถูกหารด้วย 16 หรือ 10H ยกเว้นโอเปอร์รานด์ที่แอสแซมเบลอร์สมมุติเป็น PARA

COMBINE TYPE วิธีการกำหนดแบบนี้จะเป็นการรวมเข้ากับเซคเม้นต์อื่นๆ เมื่อมีการ LINKED เข้าด้วยกัน ชนิดของ COMBINE TYPE คือ STACK,COMMON PUBLIC และ AT expression ตัวอย่างการกำหนดสแตคเซคเม้นต์ดังนี้

```
NAME        SEGMENT         PARA   STACK
```

ท่านจะใช้คำสั่ง PUBLIC และ COMMON เมื่อแยกโปรแกรมแอสแซมเบลอ แต่ถ้ารวมกันใช้ LINKED ส่วนในกรณีอื่นๆ โปรแกรมที่เขียนจะไม่รวมกับโปรแกรมอื่นๆ

CLASS TYPE จุดเข้านี้จะมีตัว APOSTROPHE S (') กำกับในการใช้กับกลุ่มของเซคเม้นต์ซึ่งสัมพันธ์กัน เมื่อมีการ LINK เช่น 'CODE' สำหรับ CODE SEGMENT

```
NAME        SEGMENT         PARA STACK  'STACK'
```

คำสั่งเทียม PROC

คำสั่งเทียม PROC ใช้ในส่วนของการกำหนด CODE SEGMENT ที่เป็นส่วนของการเอ็กซีคิวส์โปรแกรม ส่วนของ CODE SEGMENT จะมี PROCEDURE หนึ่ง PROCEDURE หรือมากกว่าหนึ่งก็ได้ การกำหนดแต่ละ PROCEDURE จะต้องใช้คำสั่งเทียม PROC

```
SEGNAME     SEGMENT   PARA    COMMENT
PROCNAME    PROC      FAR     ;ONE
                      -       ;PROCEDURE
                      -       ;WITH IN THE CODE
PROCNAME    ENDP              ;SEGMENT
SEGNAME     ENDS
```

การกำหนด PROCEDURE จะต้องมีการกำหนดชื่อ OPERAND FAR จะเป็นตัวชี้โปรแกรม DOS สำหรับโหลด PROCEDURE โดยมีคำสั่งเทียม PROC เป็นจุดเข้าโปรแกรมที่จะเอ็กซีคิวส์ ส่วนคำสั่งเทียม ENDP จะเป็นตัวกำหนดจุดสิ้นสุดของ PROCEDURE ในส่วนของ CODE SEGMENT จะมี SUB PROCEDURE ตามปกติจะใช้ OPERAND NEAR

คำสั่ง ASSUME

โปรเซสเซอร์จะใช้รีจีสเตอร์ SS กำหนดแอดเดรสของสแตค รีจีสเตอร์ DS ใช้กำหนดแอดเดรสของดาต้าเซคเม้นต์ รีจีสเตอร์ CS กำหนดแอดเดรสของ CODE SEGMENT ท่านจะต้องมีการบอกชื่อของรีจีสเตอร์เหล่านี้ให้แอสแซมเบลอร์ถึงจุดประสงของแต่ละเซคเม้นต์ดังต่อไปนี้

```
OPERATION       OPERAND
ASSUME          SS:STACKNAME,DS:DATASEGNAMEMCS:CODESEGNAME
```

แต่ในการใช้งานถ้าใช้ ES เก็บข้อมูลเราก็สามารถใช้ ES:DATASEGNAME แต่ถ้าไม่ใช้ ES เรากำหนด ES:NOTHING

คำสั่ง END

คำสั่งเทียม END จะเป็นการใช้เมื่อสิ้นสุดการเขียนโปรแกรม หรือสิ้นสุดเซคเม้นต์ใช้ ENDSหรือเป็นการสิ้นสุด PROCEDURE เราก็ใช้ ENDP

OPERATION	OPERAND
END	[PROCNAME]

MEMORY AND REGISTER REFERENCE

เราจะต้องทำความเข้าใจให้ชัดเจนเกี่ยวกับรีเฟอร์รานด์ของคำสั่งที่เป็นชื่อ ชื่ออยู่ในวงเล็บและจำนวน ตัวอย่างต่อไปนี้สมมุติ WORDA กำหนดค่าเวิร์ดในหน่วยความจำดังนี้

```
MOV     AX,BX          ;Move contents of BX to AX
MOV     AX,WORDA       ;Move contents WORDA to AX
MOV     AX,25          ;Move calue 25 to AX
MOV     AX,[BX]        ;Move contents of memory location
                       ;specified by BX
```

โครงสร้างโปรแกรม PROGRAM ORGANIZATION

โปรแกรมที่จะทำการเอ็กซีคิวส์มี 2 ชนิดคือ EXE และ COM การพัฒนาโปรแกรมจะต้องพัฒนาเป็น EXE ก่อน ในรูป 3.1 เป็นโครงสร้างโปรแกรมแบบ EXE มีการเรียงลำดับการกำหนดเซคเม้นต์ดังนี้

```
STACKSG     SEGMENT   PARA     STACK'Stack'
DATASG      SEGMENT   PARA     'Data'
CODESG      SEGMENT   PARA     'Code'
```

การกำหนดค่าเริ่มแรกของโปรแกรม INITIALLIZING A PROGRAM

รูป 3.1 แสดงโครงสร้างของโปรแกรมที่เป็น EXE จะมีการกำหนดค่าเริ่มแรก 2 ชนิดคือ (1) สั่งเกตุจากกลุ่มรีจีสเตอร์เซคเม้นต์ที่จะบอกแอสแซมเบลอร์ (2) โหลดค่าแอดเดรสลงใน DS

```
                    PAGE    60,132
                    TITLE   EXASM1 Skeleton of an assembly program
;-------------------------------------------------------------------------
STACKSG             SEGMENT PARA STACK 'Stack'
                    ----
STACKSG             ENDS
;-------------------------------------------------------------------------
DATASG              SEGMENT PARA  'Data'
                    ----
DATASG              ENDS
;-------------------------------------------------------------------------
CODESG              SEGMENT PARA 'Code'
BEGIN               PROC    FAR
                    ASSUME  CS:CODESG,DS:DATASG,SS:STACKSG
                    MOV     AX,DATASG       ;GET ADDRESS OF DATA SEGMENT
                    MOV     DS,AX           ;STORE IT IN DS
                    ---
                    MOV     AH,4CH          ;REQUEST TERMINATE
                    INT     21H             ;EXIT TO DOS
BEGIN               ENDP
CODESG              ENDS
                    END     BEGIN
```

รูป 3.1 Skeleton of an EXE Program

คำสั่ง ASSUME ใน CODE SEGMENT จะเป็นตัวบอกแอสแซมเบลอร์ถึงกลุ่มรีจีสเตอร์เซคเม้นต์ว่าชื่ออะไรเช่น CS ชื่อ CODESEG , DS ชื่อ DATASEG , และ SS ชื่อ STACKSEG

```
ASSUME  CS:CODESEG,DS:DATASEG,SS:STACKSEG
```

คำสั่ง ASSUME เป็นการจัดกลุ่มเซคเม้นต์ของรีจีสเตอร์เซคเม้นต์ แอสแซมเบลอร์สามารถกำหนดค่าออฟเซ็ทแอดเดรสในแต่ละเซคเม้นต์ ดังตัวอย่าง แต่ละคำสั่งที่อยู่ใน CODE SEGMENT ที่กำหนดความยาวคำสั่งแรกจะมีค่าออฟเซ็ทเป็น 0 และถ้าคำสั่งมีความยาว 2 ไบท์ คำสั่งที่ 2 จะมีค่าออฟเซ็ทเป็น 2 ต่อไปเรื่อยๆตามความยาวของคำสั่ง

คำสั่งที่กำหนดค่าเริ่มแรกของ DATA SEGMENT ใน DS คือ

```
MOV   AX,DATASEG          ;GET ADDRESS OF DATA SEGMENT
MOV   DS,AX               ;STORE ADDRESS IN DS
```

คำสั่ง MOV คำสั่งแรกจะทำการโหลดแอดเดรสของ DATA SEGMENT ในรีจีสเตอร์ AX และคำสั่ง MOV คำสั่งที่ 2 จะเคลื่อนย้ายข้อมูล AX ไปยัง DS จะเห็นว่ามีการเคลื่อนย้ายถึง 2 ครั้ง เพราะไม่มีคำสั่งใดที่เคลื่อนย้ายข้อมูลได้โดยตรงจากหน่วยความจำไปยังรีจีสเตอร์เซคเม้นต์ ถ้าใช้คำสั่ง MOV DS,DATASEG เป็นการคำสั่งที่ผิด ไม่สามารถใช้ได้

เมื่อ DOS จะทำการโหลดโปรแกรมเอ็กซีคิวส์ไปยังหน่วยความจำ การเอ็กซีคิวส์มีการทำงานดังนี้

1. โครงสร้าง 256 ไบท์ (100H) ของ PROGRAM SEGMENT PREFIX (PSP) จะเป็นตัวกำหนดขอบเขตของหน่วยความจำ
2. โหลดโปรแกรม EXE ทันทีตาม PSP
3. เก็บแอดเดรสของ PSP ใน DS และ ES
4. เก็บแอดเดรสของ CODE SEGMENT ใน CS และเซ็ทค่าของ IP เป็นค่าออฟเซ็ทแอดเดรสของคำสั่ง ปกติจะมีค่าเป็น 0
5. เก็บค่าแอดเดรสของสแตคใน SS และเซ็ทค่า SP ที่ยอดของสแตค
6. เคลื่อนย้ายการควบคุมไปที่โปรแกรม EXE เพื่อการเอ็กซีคิวส์

โปรแกรม DOS จะโหลดค่าเริ่มแรก CS:IP และ SS:SP แต่การโหลดโปรแกรมจะเซ็ทแอดเดรสของ PSP ใน DS และ ES ท่านจะต้องเซ็ทสภาวะเริ่มแรก ตามรูป 3.1

การสิ้นสุดการเอ็กซีคิวส์โปรแกรม TERMINATING PROGRAM EXECUTION

ในส่วนนี้จะอธิบายการสิ้นสุดของโปรแกรมในการเขียนโปรแกรมภาษาแอสแซมบลี ดังนี้

DOS SERVICE CALL 4CH

โปรแกรมภาษาแอสแซมบลี ต้องทำงานภายใต้ระบบปฏิบัติการที่เรียกว่า DOS ตั้งแต่รูป 2.0 เป็นต้นไป โดนใช้คำสั่ง INT 21H ตามรูป 3.1 โดยเก็บค่า AH = 4CH สำหรับการสิ้นสุดโปรแกรม

```
MOV  AH,4CH          ;DOS SERVICE CALL
MOV  AL,RETCODE      ;RETURN CODE
INT  21H             ;TERMINATE AND RETURN TO DOS
```

การ RETURN CODE สำหรับการสิ้นสุดโปรแกรมจะใช้ 0 อาจเขียนได้ดังนี้

```
MOV  AX,4C00H        ;DOS SERVICE CALL AND RETURN CODE
```

DOS INTERRUPT 20H

การทำงานชนิดนี้สามารถใช้กับ DOS ได้ทุกรุ่นดังนี้

```
INT    20H     ;TERMINATE AND RETURN CODE
```

DOS SERVICE CALL 0

การทำงานชนิดนี้สามารถใช้กับ DOS ได้ทุกรุ่นดังนี้

```
MOV  AH,0            ;DOS SERVICE CALL
INT  21H             ;FOR PROGRAM TERMINATION
```

คำสั่ง RET

การใช้คำสั่ง RET เป็นคำสั่งที่เซ็ทให้โปรแกรมย้อนกลับสู่ DOS เมื่อโปรแกรมเอ็กซีคิวส์เรียบร้อยแล้ว ข้อมูลที่ตามหลังคำสั่ง ASSUME จะ PUSH ค่าลงในสแตคคือ แอดเดรสของรีจีสเตอร์ DS และคำสั่ง PUSH ที่ 2 คือ ZERO ADDRESS คำสั่ง RET จะเริ่มต้นที่โปรแกรม PROGRAM SEGMENT PREFIX (PSP) เมื่อคำสั่ง INT 20H จะถูกเก็บค่าไว้

```
ASSUME    CS:CSEG,DS:DSEG,SS:SSEG,ES:DSEG
PUSH      DS                    ;PUSH DS ONTO SATCK
SUB       AX,AX                 ;SET AX TO ZERO
PUSH      AX                    ;PUSH ZERO ONTO STACK
MOV       AX,DSEG               ;INITIALIZE
MOV       DS,AX                 ;DS REGISTER
---
RET                             ;TERMINATE AND RETURN TO DOS
```

ข้อมูลของรีจีสเตอร์ DS จะเซ็ทค่าเริ่มแรกของ PSP ซึ่งการ PUSH ครั้งแรกจะเก็บค่าลงในสแตค คำสั่ง SUB จะทำการเคลียค่าใน AX เป็น 0 โดยการลบตัวเอง คำสั่ง PUSH คำสั่งที่ 2 จะเก็บค่า 0 ลงในสแตค ปัจจุบันสแตคจะมีแอดเดรส 2 ค่า

```
¦ .... ¦ 0000 ¦ address of PSP ¦
```

คำสั่ง RET จะทำหน้าที่ออกจากโปรแกรมเพื่อกลับสู่ DOS หมายความว่าแอดเดรสที่ PUSH ลงในสแตค ที่เริ่มต้น (PUSH DS) เป็นผลทำให้คำสั่ง RET จะทำการ POP ที่ยอดแอดเดรสของสแตค (ZEROS) ลงในค่า IP และเพิ่มค่าของ SP เป็น 2 มันจะทำการ POP ที่เวิร์ดต่อไปของยอดสแตค (แอดเดรสของ PSP) ลงใน CS และเพิ่มค่า SP อีก 2 จะได้ค่า CS:IP ใหม่คือ เซคเม้นต์และค่าออฟเซ็ท ที่จุดนี้เป็นจุดเริ่มต้นของ PSP คำสั่ง INT 20H จะเป็นการควบคุมจะส่งกลับไปที่ DOS

ตัวอย่างของโปรแกรม

รูป 3.2 เป็น SOURCE PROGRAM ของภาษาแอสแซมบลีในการบวกค่า 2 ค่า ในรายการข้อมูลที่รีจีสเตอร์ AX STACKSEG จะมีข้อมูล 1 ค่า กำหนดโดย DW เป็นการกำหนดค่าขนาด 32 บิต ที่มีค่าเริ่มต้นเป็น 0

DATASEG มีการกำหนดข้อมูลขนาด 3 ตัวมีชื่อ FLDA , FLRB , FLDC

CODESEG เป็นส่วนที่เก็บข้อมูลของโปรแกรมเอ็กซีคิวส์ เพราะว่าคำสั่ง ASSUME เป็นคำสั่งเทียมที่กำหนดค่า CODESEG ในรีจีสเตอร์ CS DATASEG กำหนดค่ารีจีสเตอร์ DS และ STACKSEG กำหนดค่ารีจีสเตอร์ของ SS ผลการทำงานของ ASSUME จะบอกแอสแซมเบลอร์ในการใช้แอดเดรสของรีจีสเตอร์ CS

สำหรับแอดเดรส CODESEG แอดเดรสในรีจีสเตอร์ DS สำหรับ DATASEG และแอดเดรสของรีจีสเตอร์ SS สำหรับแอดเดรสของ STACKSEG เมื่อมีการโหลดโปรแกรมจากดิสก์ลงในหน่วยความจำ สำหรับการเอ็กซีคิวส์ ระบบจะทำการโหลด ACTUAL ADDRESS ใน SS และ CS แต่ไม่มีการเซ็ทค่าเริ่มแรกของ DS และ ES

```
                        PAGE  60,132
                TITLE   EXASM1A (EXE) MOVE AND ADD OPERATIONS
;-------------------------------------------------------------------
STACKSG         SEGMENT         PARA  STACK 'Stack'
                DW              32 DUP (0)
STACKSG         ENDS
;-------------------------------------------------------------------
DATASG          SEGMENT         PARA 'Data'
     FLDA       DW         250
     FLDB       DW         125
     FLDC       DW         ?
DATASG          ENDS
;-------------------------------------------------------------------
CODESG          SEGMENT         PARA 'Code'
BEGIN           PROC            FAR
                ASSUME  CS:CODESG,DS:DATASG,SS:STACKSG
            MOV     AX,DATASG      ;SET ADDRESS OF DATASG
            MOV     DS,AX          ;IN DS REGISTER
            MOV     AX,FLDA        ;MOVE 250 TO AX
            ADD     AX,FLDB        ;ADD 125 TO AX
            MOV     FLDC,AX        ;STORE SUM IN FLDC
            MOV     AX,4C00H       ;RETURN TO DOS
            INT     21H
BEGIN       ENDP                   ;END OF PROCEDURE
CODESG      ENDS
            END     BEGIN          ;END OF PROGRAM
```

รูป 3.2 EXE Source Program with Conventional Segment

INITIALZING FOR 80386 PROTECTED MODE

สำหรับการทำงานของ PROTECTED MODE ภายใต้การทำงานของโปรเซสเซอร์ 80386 / 80486 จะใช้ DWORD ในการกำหนด ALIGN SEGMENTS ในแอดเดรส DOUBLEWORD เพราะความเร็วในการเข้าถึงหน่วยความจำขนาด 32 บิตในบัสข้อมูล ในตัวอย่างต่อไปนี้จะใช้คำสั่งเทียม .386 บอกแอสแซมเบลอร์ให้รับรู้ว่าโปรแกรมทำงานภายใต้ 80386

คำสั่ง USE32 จะบอกแอสแซมบลอร์ในการทำงาน 32 บิต ในส่วนของ PROTECTED MODE

```
        .386
SEGNAME     SEGMENT     DWORD  USE32
```

การกำหนดค่าเริ่มแรกของรีจีสเตอร์ DS (DATA SEGMENT) มีลักษณะเหมือนกันแต่ 80386 ยังใช้รีจีสเตอร์ DS ขนาด 16 บิต

```
        MOV   EAX,DATASEG       ;GET ADDRESS OF DATA SEGMENT
        MOV    DS,AX            ;LOAD 16 BIT PORTION
```

SIMPLIFIES SEGMENT DIRECTIVES

แอสแซมเบลอร์ 5.0 ของไมโครซอฟท์และ BORLAND TURBO ASSEMBLER จะมีการกำหนดเซคเม้นต์ใน MEMORY MODELS ก่อนการใช้งานทุกๆครั้ง ทุกๆเซคเม้นต์ มีรูปแบบดังนี้

```
.MODEL    MEMORY _ MODEL
```

MODEL	NUMBER OF CODE SEGMENT	NUMBER OF DATA SEGMENT
SMALL	1	1
MEDUIM	MORE THAN 1	1
COMPACT	1	MORE THAN 1
LARGE	MORE THAN 1	MORE THAN 1

CT 215 61

MEMORY MODEL อาจจะเป็น SMALL , MEDIUM , CPMPACT หรือ LARGE ในตาราง ท่านสามารถใช้โมเดลเหล่านี้ในโปรแกรมได้เช่น SMALL เป็นโมเดลของรหัสคำสั่งที่ใช้เพียง 64 K และในส่วนของข้อมูล 64 K ตัวอย่างในหนังสือจะใช้เพียง SMALL คำสั่งเทียม MODEL จะทำงานร่วมกับ ASSUME โดยอัตโนมัติ รูปแบบทั่วๆไปของคำสั่งเทียมในการกำหนดค่าของ CODE SEGMENT , DATA SEGMENT และ STACK SEGMENT มีดังนี้

```
.CODE  (NAME)
.DATA
.STACK  (SIZE)
```

สำหรับคำสั่งเทียมแต่ละคำสั่ง ที่กำหนดค่าแอดเดรสแต่ละเซคเม้นต์โดยใช้คำสั่ง SEGMENT และทำร่วมกับคำสั่ง ENDS การกำหนดชื่อเซคเม้นต์คือ _TEXT สำหรับ CODE SEGMENT _DATA และ _STACK การใช้ขีดเส้นใต้ (UNDERLINE หรือ BREAK) ที่นำหน้า _TEXT , _DATA เป็นชื่อของเซคเม้นต์แต่ละชนิด ท่านสามารถกำหนด 3 เซคเม้นต์ให้แอสแซมเบลอร์ คำสั่งที่กำหนดค่าเริ่มแรกใน DATA SEGMENT คือ

```
MOV  AX,@DATA
MOV  DS,AX
```

จากรูป 3.2 เราสามารถเปลี่ยนการเขียนโปรแกรมให้สะดวกในการกำหนดเซคเม้นต์ดังในตัวอย่างรูป 3.3 โดยใช้การกำหนดเซคเม้นต์ .STACK .DATA และ .CODE โดยไม่ต้องมีคำสั่ง SEGMENT ENDS และ ASSUME

```
                PAGE  60,132
        TITLE   EXASM1A (EXE) MOVE AND ADD OPERATIONS
;-------------------------------------------------------------------
        .MODEL  SMALL
        .STACK  64              ;DEFINE STACK
        .DATA                   ;DEFINE DATA
FLDA    DW      250
FLDB    DW      125
FLDC    DW      ?
```

```
                .CODE                   ;DEFINE CODE SEGMENT
BEGIN           PROC    FAR
                MOV     AX,@DATA        ;SET ADDRESS OF DATASG
                MOV     DS,AX           ;IN DS REGISTER
                MOV     AX,FLDA         ;MOVE 250 TO AX
                ADD     AX,FLDB         ;ADD 125 TO AX
                MOV     FLDC,AX         ;STORE SUM IN FLDC
                MOV     AX,4C00H        ;RETURN TO DOS
                INT     21H
BEGIN           ENDP                    ;END OF PROCEDURE
                END     BEGIN           ;END OF PROGRAM
```

รูป 3.3 EXE Source Program with Simplified Segment Directives

คำสั่ง อินพุตเอ้าพุต

เราได้ศึกษาโครงสร้างโปรแกรมทั้ง 2 แบบที่ใช้ในการเขียนโปรแกรมภาษาแอสแซมบลี และเราได้ศึกษาว่าหน่วยประมวลผลกลาง (CPU) ทำหน้าที่ติดต่อกับอุปกรณ์รอบข้าง ส่งผ่านทางรีจีสเตอร์อินพุตเอ้าพุต ที่เรียกว่า I/O PORTS คำสั่งการทำงานของ I/O PORTS มี 2 ชนิดคือ IN และ OUT ซึ่งเป็นการเข้าถึงข้อมูลโดยตรงจากพอร์ต ซึ่งคำสั่งเหล่านี้จะใช้เพื่อต้องการความเร็วในการทำงาน เช่น โปรแกรมเกมส์ อย่างไรก็ตามโปรแกรมประยุกต์ใช้งานส่วนมากจะไม่ใช้คำสั่ง IN และ OUT เพราะว่า 1) แอดเดรสพอร์ตขึ้นอยู่กับโมเดลของคอมพิวเตอร์ 2)ถ้าใช้โปรแกรมอินพุตเอ้าพุตของการบริการจะง่ายกว่าที่เขียนโดยโรงงานผู้ผลิต

โปรแกรมอินพุตเอ้าพุตของการบริการมี 2 ชนิดคือ 1) BASIC INPUT OUTPUT SYSTEM (BIOS) 2) DISK OPERATING SYSTEM (DOS) โปรแกรมไบออสจะเป็นโปรแกรมที่อยู่ในรอมและมีหน้าที่โดยตรงกับ I/O PORTS จะได้อธิบายต่อไป เราสามารถใช้คำสั่งพื้นฐานในการทำงานของจอภาพ เช่นการเคลื่อนย้ายเคอร์เซอร์ และการเลื่อนจอภาพ ส่วนการบริการของดอส ใช้ในส่วนที่ยุ่งยากกว่า เช่นการพิมพ์ตัวอักขระ

คำสั่ง INT

การใช้คำสั่งของ DOS และ BIOS เราจะใช้คำสั่ง INT มีรูปแบบดังต่อไปนี้

```
INT     Interrupt_number
```

ขณะที่ Interrupt_number คือจำนวนที่กำหนดการทำงาน ตัวอย่าง คำสั่ง INT 16H ซึ่งเป็นคำสั่งที่ใช้ในการบริการของ BIOS ทำหน้าที่รับข้อมูลจากคีย์บอร์ด เป็นต้น

INT 21H

คำสั่ง INT 21H ซึ่งเป็นคำสั่งการบริการของ DOS จำนวนที่กำหนดการทำงานต่างๆ ที่เรียกว่า Function call จะเก็บไว้ใน AH และตามด้วยคำสั่ง INT 21H ดังตัวอย่างที่น่าสนใจดังต่อไปนี้

FUNCTION NUMBER	ROUTINE
1	SINGLE-KEY INPUT
2	SINGLE-CHAR OUTPUT
9	CHARACTER STRING OUTPUT

การทำงานของการบริการ 1

```
INPUT    AH  = 1
OUTPUT   AL  = ASCII CODE (ถ้ามีการพิมพ์ตัวอักขระ)
             = 0 (ไม่มีการพิมพ์)
```

เราสามารถเขียนคำสั่งได้ดังนี้

```
MOV  AH,1       ;INPUT KEY FUNCTION
INT  21H        ;ASCII CODE IN AL
```

การทำงานของคำสั่งนี้จะเป็นการรอคอยผู้ใช้ให้กดคีย์บอร์ด ถ้าผู้ใช้กดคีย์บอร์ด ข้อมูลรหัส ASCII จะรับค่าเก็บไว้ใน AL และแสดงผลที่จอภาพ ถ้ามีการกดคีย์อื่นๆ เช่นลูกศร F1 - F10 และอื่นๆค่าใน AL จะเป็น 0 คำสั่ง INT 21H จะทำงานตามข้อมูลที่รับมาใน AL

การทำงานของคำสั่ง INT 21H การบริการที่ 1 จะไม่มี Prompt ให้ผู้ใช้ในการป้อนข้อมูล ทำให้ผู้ใช้อาจไม่รู้ว่าคอมพิวเตอร์กำลังรอรับข้อมูลอยู่ คำสั่งต่อไปนี้จะเป็นตัวสร้าง Prompt

การทำงานของการบริการ 2

INPUT	AH	=	2
	DL	=	ASCII CODE (การพิมพ์ตัวอักขระทีละตัวหรือตัวควบคุม)
OUTPUT	AL	=	ASCII CODE (แสดงตัวอักขระหรือตัวควบคุมที่ละตัว)

เราสามารถเขียนคำสั่งได้ดังนี้

```
MOV  AH,2       ;DISPLAY CHARACTER
MOV  DL,'?'     ;CHARACTER IS '?'
INT  21H        ;DISPLAY CHARACTER
```

หลังจากการแสดงตัวอักขระแล้ว เคอร์เซอร์จะเลื่อนไปอีก 1 ตำแหน่ง ถ้าเป็นตัวอักขระควบคุม ข้อมูลที่อยู่ใน DL จะอยู่ในรูปรหัส ASCII เมื่อเอ็กซีคิวส์คำสั่ง INT 21H ก็จะทำงานตามการควบคุมตามรหัสที่กำหนด จากคำสั่งที่เขียนขึ้น 3 คำสั่งข้างบนเราจะเห็นว่า เครื่องหมาย '?' จะแสดงบนจอภาพ

ASCII	SYMBOL	FUNCTION
7	BEL	BEEP (SOUNDS A TONE)
8	BS	BACKSPACE
9	HT	TAB
A	LF	LINE FEED
D	CR	CARRIAGE RETURN

โปรแกรมแสดงการทำงานของการบริการ

การเขียนโปรแกรมในการอ่านข้อมูลจากคีย์บอร์ดและแสดงผลในบรรทัดต่อไป เราเริ่มต้นจากการแสดงผลเครื่องหมาย '?'

```
MOV  AH.2       ;DISPLAY CHARACTER FUNCTION
MOV  DL,'?'     ;CHARACTER IS '?'
INT  21H        ;DISPLAY CHARACTER
```

หลังจากจอภาพแสดงเครื่องหมาย Prompt '?' แล้ว ให้อ่านข้อมูลโดยใช้คำสั่งดังนี้

```
MOV  AH,1       ;READ CHAR FUNCTION
INT  21H        ;CHAR IN AL
```

หลังจากรับข้อมูลและให้แสดงผลบรรทัดต่อไป ก่อนที่จะแสดงผลตัวอักขระใน AL จะต้องเก็บไว้ก่อนดังนี้

```
MOV  BL,AL      ;SAVE IT IN BL
```

การเคลื่อนย้ายเคอร์เซอร์ไปเริ่มต้นที่บรรทัดใหม่จะต้องเอ็กซีคิวส์คำสั่ง CR และ LF ซึ่งมีรูปแบบการทำงานต่อไปนี้

```
MOV  AH.2
MOV  DL,0DH          ;CRRIAGE RETURN
INT  21H
MOV  DL,0AH          ;LINE FFED
INT  21H
```

หลังจากนั้นก็ให้นำค่าของ AL ที่เก็บใน BL ออกมาใช้การบริการที่ 2 และตามด้วยคำสั่ง INT 21H ดังนี้

```
        MOV   DL,BL             ;GET CHARACTER
        INT   21H               ;AND DIAPLAY IT
```

เป็นการสิ้นสุดการทำงานของโปรแกรมที่สมบูรณ์ ดังการเขียนโปรแกมต่อไปนี้

```
        TITLE     SAMPLE PROGRAM
        .MODEL SMALL
        .STACK 100H
        .CODE
MAIN    PROC
    ;DISPLAY PROMPT
        MOV   AH,2        ;DISPLAY CHARACTER FUNCTION
        MOV   DL,'?'      ;CHARACTER IS '?'
        INT   21H         ;DISPLAY CHARACTER
    ;INPUT A CHARACTER
        MOV   AH,1        ;READ CHAR FUNCTION
        INT   21H         ;CHAR IN AL
        MOV   BL,AL       ;SAVE IT IN BL
    ;GOTO A NEW LINE
        MOV   AH,2
        MOV   DL,0DH          ;CRRIAGE RETURN
        INT   21H
        MOV   DL,0AH          ;LINE FFED
        INT   21H
    ;DISPLAY CHARACTER
        MOV   DL,BL           ;GET CHARACTER
        INT   21H             ;AND DIAPLAY IT
    ;RETURN TO DOS
        MOV   AH,4CH
        INT   21H
    MAIN ENDP
        END   MAIN
```

บทสรุป

- เซมิโคลอน (;) ใช้นำหน้าบรรทัดหมายเหตุ
- โปรแกรมจะประกอบด้วย 1 หรือมากกว่า 1 เซคเม้นต์ แต่ละชนิดของเซคเม้นต์มีจุดเริ่มต้นของต้นเอง
- ENDS เป็นจุดสิ้นสุดของแต่ละเซคเม้นต์ ENDP เป็นจุดสิ้นสุดของ Procedure END เป็นจุดสิ้นสุดของโปรแกรม
- คำสั่ง ASSUME เป็นการกำหนดชื่อเซคเม้นต์ให้กับรีจีสเตอร์เซคเม้นต์แต่ละตัวคือ CS DS และ SS
- โปรแกรม EXE จะต้องกำหนดสแตคอย่างน้อย 32 เวิร์ด
- การกำหนดขนาดของโปรแกรมมีดังนี้ SMALL,MEDIUM,COMPACT และ LARGE

แบบฝึกหัด

3.1 คำสั่งอะไรของแอสแซมเบลอร์ที่ทำหน้าที่ดังต่อไปนี้

a. พิมพ์ที่หัวกระดาษสูงสุดของรายการโปรแกรม

b. เลื่อนกระดาษขึ้นหน้าใหม่

3.2 จงอธิบายถึงจุดประสงค์ของแต่ละเซคเม้นต์ในการเขียนโปรแกรมภาษาแอสแซมบลี

3.3 จงอธิบายความหมายของคำสั่งเทียมและคำสั่งภาษาแอสแซมบลีต่างกันอย่างไร

3.4 รูปแบบของการกำหนดเซคเม้นต์แบบนี้หมายความว่าอย่างไร

NAME SEGMENT ALIGN COMBINE 'CLASS'

3.5 จุดประสงค์ในการกำหนด PROCEDURE มีจุดเริ่มต้นอย่างไร และสิ้นสุดอย่างไร

3.6 มีวิธีการกำหนดจุดเริ่มต้นและกำหนดจุดสุดท้ายของ Procedure อย่างไร

3.7 Procedure Far และ Near ต่างกันอย่างไร

3.8 จงยกตัวอย่างการกำหนดชื่อให้กับรีจีสเตอร์เซคเม้นต์แต่ละตัว

3.9 โปรแกรม EXE ต่างกับโปรแกรม COM อย่างไร
