บทที่ 7 โปรแกรมภาษาแอสเซมบลี้ (1)

- รูปแบบของโปรแกรมภาษาแอสเซมบลี้
- รูปแบบของโปรแกรมภาษาแอสเซมบลี้แบบใหม่
- การเรียกใช้บริการของ DOS
- ขั้นตอนการแปลโปรแกรม
- ตัวอย่างโปรแกรม

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

รูปแบบของโปรแกรมภาษาแอสเซมบลี้

- เราจะเขียนโปรแกรมซึ่งมีลักษณะเป็นเซกเมนต์หลาย ๆ เซกเมนต์. ในการเขียนเราจะประกาศเซกเมนต์ต่าง ๆ และกำหนดข้อมูลและโปรแกรมลงในเซกเมนต์ต่าง ๆ เหล่านั้น

EX

```
;
; This program prints the message ”Hello world”
;
dseg    segment
msg1    db    ’Hello world’,10h,13h,’$’
dseg    ends

sseg    segment stack
        db    100 dup (?)
sseg    ends

cseg    segment
        assume cs:cseg,ds:dseg,ss:sseg
start:
        mov   ax,dseg         ;set DS
        mov   ds,ax
        mov   ah,9h           ;print message
        mov   dx,offset msg1
        int   21h
        mov   ax,4c00h        ;exit program
        int   21h
cseg    ends
        end   start
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

รูปแบบโปรแกรม

- **คำสั่งเทียม**

คำสั่งเทียม คือ คำสั่งที่ผู้เขียนโปรแกรมเขียนเพื่อระบุให้ assembler แปลโปรแกรมในรูปแบบที่ต้องการ. คำสั่งกลุ่มนี้จะไม่ปรากฏในรหัสคำสั่งภาษาเครื่องที่แปลเรียบร้อยแล้ว. ตัวอย่างคำสั่งเทียม เช่น **segment**, **db**, และ **assume** เป็นต้น.

- **การประกาศเซกเมนต์**

ในโปรแกรมภาษาแอสเซมบลี้เราสามารถประกาศเซกเมนต์ได้โดยใช้**คำสั่งเทียม segment**.

segment_name **segment**

....

segment_name **ends**

จากโปรแกรมตัวอย่าง เราได้ประกาศเซกเมนต์ทั้งสิ้น 3 เซกเมนต์ คือ **dseg sseg** และ **cseg**.

รูปแบบโปรแกรม

- **คำสั่งเทียม `assume` : ประกาศให้ assembler ทราบการใช้เซกเมนต์**

เราใช้ **คำสั่งเทียม `assume`** เพื่อระบุให้ assembler ทราบว่าเราจะใช้เซกเมนต์ต่าง ๆ อย่างไร.

การระบุโดยวิธีนี้นั้นจะเป็นการระบุให้ assembler นำไปแปลโปรแกรมได้ถูกต้องเท่านั้น ไม่ได้ระบุให้ assembler ตั้งค่าเซกเมนต์รีจิสเตอร์ต่างๆ ให้. ดังนั้นเราจะต้องตั้งค่าให้กับเซกเมนต์รีจิสเตอร์เอง.

EX

```
mov     ax,dseg
mov     ds,ax
```

- **ข้อยกเว้น : เซกเมนต์คำสั่ง และแสต็กเซกเมนต์**

โดยปกติแล้วระบบปฏิบัติการจะตั้งค่าของ CS และ SS ให้กับโปรแกรมเมื่อเริ่มทำงาน. เราจะต้องระบุเซกเมนต์ที่จะใช้เป็น stack โดยใช้**คำสั่งเทียม `stack`** หลังการประกาศเซกเมนต์ที่ต้องการให้เป็นแสต็ก.

กรณีของ CS ระบบจะตั้งให้อัตโนมัติ

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

รูปแบบโปรแกรม

- **การประกาศจุดเริ่มโปรแกรม**

เราจะระบุจุดเริ่มต้นโปรแกรมหลัง**คำสั่งเทียม end** ในบรรทัดสุดท้าย. ในการระบุตำแหน่งนั้นเราจะระบุเป็น label.

- **การประกาศเลเบล**

เราสามารถระบุตำแหน่งของหน่วยความจำได้โดยการสร้างเลเบลที่ตำแหน่งนั้น.

label_name **:**

Assembler จะจดจำแอดเดรสของเลเบลต่าง ๆ และจะนำไปแทนค่าให้ตามความเหมาะสม

- **การใส่หมายเหตุ**

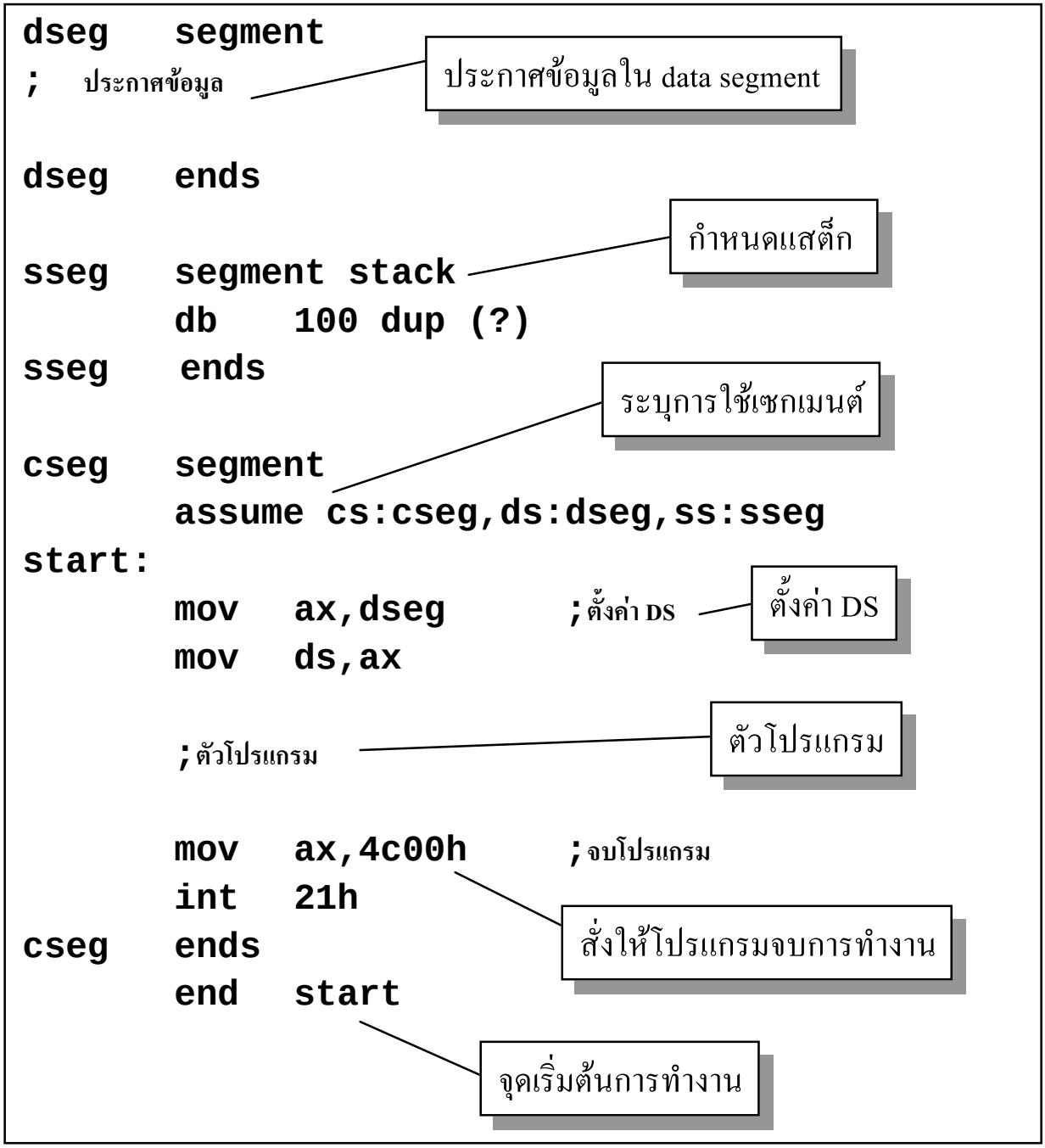
หลังเครื่องหมาย ‘ ; ’ assembler จะถือว่าเป็นหมายเหตุ

- **การสั่งให้โปรแกรมจบการทำงาน**

เราจะเรียกใช้บริการหมายเลข 4Ch ของระบบปฏิบัติการ DOS โดยใช้คำสั่ง :

```
mov    ax,4C00h
int    21h
```

ตัวอย่างโครงร่างของโปรแกรม



สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

รูปแบบโปรแกรมแบบใหม่

โปรแกรม Assembler ในปัจจุบันได้รองรับการเขียนโปรแกรมในรูปแบบใหม่ที่มีความกระทัดรัดขึ้น โดยโปรแกรมได้กำหนด MACRO ในการประการโครงร่างของโปรแกรมแบบใหม่.

```
;
; This program prints "Hello world"
;
.model  small
.dosseg

.data
msg1    db   'Hello world',10h,13h,'$'

.stack  100h

.code
start:
        mov  ax,@data
        mov  ds,ax
        mov  ah,9h
        mov  dx,offset msg1
        int  21h
        mov  ax,4c00h
        int  21h
end     start
```

ชื่อของ segment จะถูกประกาศให้โดยอัตโนมัติ. สังเกตว่าเราจะใช้ชื่อของเซกเมนต์ข้อมูลว่า @data.

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

การเรียกใช้บริการของ DOS

ระบบปฏิบัติการ DOS ได้จัดเตรียมบริการต่าง ๆ ให้ผู้เขียนโปรแกรมเรียกใช้ได้โดยผ่านทางการขัดจังหวะหมายเลข 21h.

ในการเรียกใช้บริการของ DOS เราจะต้องกำหนดหมายเลขของบริการลงในรีจิสเตอร์ AH และกำหนดค่าพารามิเตอร์ต่างๆ ลงในรีจิสเตอร์ที่ถูกกำหนดไว้. จากนั้นเราจึงเรียกใช้คำสั่ง INT 21h เพื่อเรียกใช้การบริการของระบบ.

รูปแบบโดยทั่วไปในการเรียกใช้บริการ

```
mov     ah,function_number
;
;  set parameters
;
int     21h
```

บริการของ DOS

- **Function 01h : อ่านการกดปุ่มจากแป้นพิมพ์**

  Input **AH** = 01h

  Output **AL** = รหัสแอสกีของอักขระที่ผู้ใช้กด

- **Function 02h : แสดงตัวอักษรออกทางหน้าจอ**

  Input **AH** = 02h

  DL = รหัสแอสกีของอักขระที่ต้องการแสดง

- **Function 05h : พิมพ์ตัวอักษรทางเครื่องพิมพ์**

  Input **AH** = 05h

  DL = รหัสแอสกีของอักขระที่ต้องการพิมพ์

- **Function 07h : อ่านการกดแป้นพิมพ์ โดยไม่แสดงปุ่มที่กด (ไม่ตรวจการกด Ctrl-Break)**

  Input **AH** = 07h

  Output **AL** = รหัสแอสกีของอักขระที่ผู้ใช้กด

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

บริการของ DOS

- **Function 08h : อ่านการกดแป้นพิมพ์ โดยไม่แสดงปุ่มที่กด (ตรวจการกด Ctrl-Break)**

  Input **AH** = 08h

  Output **AL** = รหัสแอสกีของอักขระที่ผู้ใช้กด

- **Function 09h : แสดงข้อความทางหน้าจอ**

  Input **AH** = 09h

  DS:DX = ตำแหน่งของข้อความที่ต้องการแสดง โดยข้อความนี้ต้องจบด้วยอักษร '$'

- **Function 0Ah : อ่านข้อความ**

  Input **AH** = 0Ah

  DS:DX = ตำแหน่งของบัฟเฟอร์สำหรับเก็บข้อมูล. (รายละเอียดของฟังก์ชั่นนี้จะกล่าวในบทถัดไป.)

- **Function 4Ch : จบโปรแกรม**

  Input **AH** = 4Ch

  AL = ค่าที่ต้องการคืนให้กับระบบ

ขั้นตอนการพัฒนาโปรแกรม

- สร้างโปรแกรมเก็บไว้ในแฟ้มนามสกุล ASM
- ใช้โปรแกรม assembler เช่น MASM หรือ TASM แปลโปรแกรมเป็นแฟ้มเป้าหมาย (Object file) โดยใช้คำสั่ง

  MASM *filename*;

```
A:\>masm ex1;
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1991.  All rights reserved.

 Invoking: ML.EXE /I. /Zm /c /Ta ex1.asm

Microsoft (R) Macro Assembler Version 6.00
Copyright (C) Microsoft Corp 1981-1991.  All rights reserved.

 Assembling: ex1.asm
```

- ใช้โปรแกรม LINK เพื่อเชื่อมโยงแฟ้มเป้าหมายแฟ้มเดียวหรือหลายแฟ้มเข้าด้วยกัน โดยใช้คำสั่ง

  LINK *filename*;

```
A:\>link ex1;

Microsoft (R) Segmented-Executable Linker  Version 5.13
Copyright (C) Microsoft Corp 1984-1991.  All rights reserved.
```

โปรแกรมตัวอย่าง

- **ตัวอย่างที่ 1**

อ่านตัวอักษรจากผู้ใช้แล้วแสดงตัวอักษรนั้นออกมา

```
;Ex1
.model  small
.dosseg
.stack  100h
.code
start:
        mov     ah,01h   ;read character (Func 01h)
        int     21h

        mov     dl,al    ;copy character to DL
        mov     ah,02h   ;display it (Func 02h)
        int     21h

        mov     ax,4C00h ;Exit (Function 4Ch)
        int     21h
end     start
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

โปรแกรมตัวอย่าง

- **ตัวอย่างที่ 2**

อ่านตัวอักษรจากผู้ใช้แล้วแสดงตัวอักษรตัวถัดไปออกมา

```
;Ex2
sseg    segment stack
        db      100 dup (?)
sseg    ends

cseg    segment
        assume  cs:cseg,ss:sseg
start:
        mov     ah,01h    ;read character (Func 01h)
        int     21h

        mov     dl,al     ;copy to DL
        inc     dl        ;increse DL (next char.)
        mov     ah,02h    ;display it (Func 02h)
        int     21h

        mov     ax,4C00h  ;Exit
        int     21h
cseg    ends
        end     start
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

โปรแกรมตัวอย่าง

- **ตัวอย่างที่ 3**

อ่านตัวอักษรพิมพ์เล็กจากผู้ใช้แล้วแสดงตัวอักษรพิมพ์ใหญ่

```
.model  small
.dosseg

.stack  100h

.code
start:
        mov     ah,01h    ;read char.
        int     21h

        mov     dl,al
        sub     dl,32     ;change char. case
        mov     ah,02h    ;display it
        int     21h

        mov     ax,4C00h  ;exit
        int     21h
end     start
```

หมายเหตุ การแปลงอักษรพิมพ์เล็กเป็นพิมพ์ใหญ่ทำได้โดยนำรหัสแอสกีมาลบด้วย 32. [สังเกตตาราง ASCII]

โปรแกรมตัวอย่าง

- **ตัวอย่างที่ 4**

อ่านตัวอักษรพิมพ์เล็กจากผู้ใช้แล้วแสดงตัวอักษรพิมพ์ใหญ่ โดยไม่แสดงอักษรที่ผู้ใช้กดให้เห็น.

```
sseg    segment stack
        db      100 dup (?)
sseg    ends

cseg    segment
        assume  cs:cseg,ss:sseg
start:
        mov     ah,08h      ;readchar(Func 08h)
        int     21h

        mov     dl,al       ;Change case
        sub     dl,32
        mov     ah,02h
        int     21h

        mov     ax,4C00h    ;exit
        int     21h
cseg    ends
        end     start
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล