แผนการสอนประจำสัปดาห์ที่ 7

หัวข้อเรื่อง โปรแกรมภาษาแอสแซมบลีเบื้องต้น
(Assembly Language Programming)

รายละเอียด

ศึกษาการเขียนโปรแกรมภาษาแอสแซมบลีแบบเต็มรูปแบบ นั่นคือจะเขียนโปรแกรมภาษาแอสแซมบลีที่เป็นโปรแกรมที่ทำงานได้จริง มีการกำหนดรูปแบบต่างๆ ครบถ้วน โปรแกรมภาษาแอสแซมบลีที่จะเขียนต่อไปนี้ไม่ได้ทำงานบนโปรแกรม DEBUG เท่านั้น จะต้องใช้ assembler แปลโปรแกรมที่เขียนขึ้นให้อยู่ในภาพที่คอมพิวเตอร์สามารถนำไปประมวลผลได้เสียก่อน

จำนวนชั่วโมงที่สอน 3 ชั่วโมง/สัปดาห์

กิจกรรมการเรียนการสอน

1. บรรยาย
2. สืบเสาะหาความรู้
3. ค้นคว้าเพิ่มเติม
4. ตอบคำถาม

สื่อการสอน

1. สื่ออิเล็กทรอนิกส์
2. เพาเวอร์พอยต์ พรีเซนเตชัน
3. บทเรียนออนไลน์
4. เอกสารอ้างอิงประกอบการค้นคว้า

แผนการประเมินผลการเรียนรู้

1. ผลการเรียนรู้

1.1 สังเกตจากงานที่กำหนดให้ไปทำมาส่ง

1.2 สังเกตจากการตอบคำถาม

1.3 สังเกตจากการนำความรู้ไปใช้

2. วิธีการประเมินผลการเรียนรู้

2.1 ตรวจผลงานภาคปฏิบัติ

2.2 ตรวจรายงาน

2.3 ตรวจแบบฝึกหัด

3. สัดส่วนของการประเมิน

3.1 ใบงานที่นักศึกษาทำมาส่ง

3.2 คะแนนเก็บในชั้นเรียน

3.3 การเข้าชั้นเรียน

เนื้อหาที่สอน

ในสัปดาห์ที่ 7 การจัดการเรียนการสอน จะเกี่ยวข้องกับรูปแบบของโปรแกรมภาษาแอสแซมบลีแบบเก่า เปรียบเทียบกับรูปแบบของโปรแกรมภาษาแอสแซมบลีแบบใหม่ ขั้นตอนการแปลโปรแกรม การเรียกใช้บริการของ DOS ตารางรหัสแอสกี ซึ่งการเขียนโปรแกรมภาษาแอสแซมบลีแบบเต็มรูปแบบ นั่นคือจะเขียนโปรแกรมภาษาแอสแซมบลีที่เป็นโปรแกรมที่ทำงานได้จริง มีการกำหนดรูปแบบต่างๆ ครบถ้วน โปรแกรมภาษาแอสแซมบลีที่จะเขียนต่อไปนี้ไม่ได้ทำงานบนโปรแกรม DEBUG เท่านั้น จะต้องใช้ assembler แปลโปรแกรมที่เขียนขึ้นให้อยู่ในภาพที่คอมพิวเตอร์สามารถนำไปประมวลผลได้เสียก่อน

7.1 รูปแบบของโปรแกรมภาษาแอสแซมบลีแบบเก่า

โปรแกรมที่ทำงานในเครื่องคอมพิวเตอร์ซึ่งใช้หน่วยประมวลผลตระกูล 80x86 นั้นจะมีการแบ่งโปรแกรมทั้งหมดเป็นเซกเมนต์ย่อย ๆ เช่น Code segment Data segment หรือ Stack segment ดังนั้นในโปรแกรมภาษาแอสแซมบลีที่เขียนจะประกอบไปด้วยเซกเมนต์ต่าง ๆ เช่นเดียวกัน ภายในเซกเมนต์ต่าง ๆ ที่ประกาศจะระบุข้อมูลและโปรแกรมที่จะอยู่ในเซกเมนต์นั้น

ตัวอย่างโปรแกรม

```
;
; 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
        mov     ds,ax
        mov     ah,9h
        mov     dx,offset msg1
        int     21h
        mov     ax,4c00h
        int     21h
cseg    ends
        end     start
```

จากตัวอย่าง จะสังเกตได้ว่าโปรแกรมได้ประกาศเซกเมนต์ทั้งหมด 3 เซกเมนต์ คือ cseg dseg และ sseg เซกเมนต์ดังกล่าวนี้ถูกประกาศด้วย**คำสั่งเทียม segment** การที่เรียกคำสั่ง segment ว่าคำสั่งเทียมเพราะคำสั่งนี้เป็นคำสั่งที่ผู้เขียนโปรแกรมระบุให้โปรแกรม assembler แปลโปรแกรมตามลักษณะที่กำหนด โดยจะไม่มีคำสั่งภาษาเครื่องถูกสร้างจากคำสั่งกลุ่มนี้ ตัวอย่างอื่น ๆ ของคำสั่งเทียมคือ db assume และ org เป็นต้น

การประกาศเซกเมนต์

การประกาศเซกเมนต์ในโปรแกรมภาษาเอสเซมบลี ใช้คู่คำสั่งเทียม segment และ ends โดยมีลักษณะการประกาศดังนี้

segment_name segment

...

segment_name ends

จากตัวอย่างได้ประกาศเซกเมนต์ cseg dseg และ sseg **คำสั่งเทียม stack** ระบุให้ระบบใช้เซกเมนต์ sseg เป็นแสต็กของโปรแกรม **คำสั่งเทียม assume** เป็นการระบุให้ assembler ได้ทราบว่าเซกเมนต์ที่ประกาศนั้นจะให้ระบบพิจารณาว่าทำหน้าที่อะไรและมีเซกเมนต์รีจิสเตอร์ใดเป็นตัวเก็บค่าเซกเมนต์ จากตัวอย่างประกาศให้ assembler ทราบว่าเซกเมนต์ cseg จะชี้โดยรีจิสเตอร์ CS เซกมนต์ dseg จะชี้โดย รีจิสเตอร์ DS และเซกเมนต์ sseg จะชี้โดยรีจิสเตอร์ SS คำสั่งเทียม assume นี้จะเป็นการบอก assembler ให้พิจารณาตามที่ระบุเท่านั้น ไม่ได้เป็นการสั่งให้ assembler กำหนดค่าต่าง ๆ ให้โดยอัตโนมัติ สังเกตได้จากในตอนต้นของโปรแกรมมีชุดคำสั่งเพื่อปรับค่าของรีจิสเตอร์ DS ดังนี้

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

ชุดคำสั่งนี้จะปรับค่าของรีจิสเตอร์ DS ให้ชี้ไปที่ dseg สำหรับรีจิสเตอร์ SS ระบบจะปรับค่าให้ชี้ไปที่เซกเมนต์ที่ระบุไว้โดยคำสั่งเทียม stack ส่วนกรณีของรีจิสเตอร์ CS นั้นระบบจะตั้งค่าให้ตรงกับเซกเมนต์ที่เริ่มต้นโปรแกรม

โปรแกรมภาษาแอสแซมบลีจะประกอบไปด้วยการประกาศเซกเมนต์ต่าง ๆ และจะสิ้นสุดโปรแกรมที่**คำสั่งเทียม end** หลังคำสั่งเทียม end จะระบุจุดเริ่มต้นของโปรแกรม ในโปรแกรมตัวอย่างระบุจุดเริ่มต้นของโปรแกรมที่เลเบล start ที่ประกาศไว้ที่ตอนต้นของโปรแกรม การประกาศเลเบลสามารถทำได้ดังนี้

label_name:

ระบบจะจดจำตำแหน่งของเลเบลที่ประกาศไว้และจะนำแอดเดรสของเลเบลไปแทนที่ให้โดยอัตโนมัติ การที่โปรแกรม assembler จัดการเรื่องเกี่ยวกับเลเบลในโปรแกรมภาษาแอสแซมบลีนั้น นับเป็นการเพิ่มความสะดวกให้กับผู้เขียนโปรแกรมเป็นอย่างมาก

การประกาศข้อมูล

ภายในเซกเมนต์ข้อมูลสามารถประกาศข้อมูลต่าง ๆ ได้จากโปรแกรมตัวอย่างประกาศข้อมูลเป็นข้อความที่จะให้โปรแกรมพิมพ์ออกมา จะศึกษารูปแบบการประกาศข้อมูลในบทต่อไป

การใส่หมายเหตุ

หลังเครื่องหมาย **';'** assembler จะตีความว่าเป็นหมายเหตุ การใส่หมายเหตุจะช่วยทำให้โปรแกรมอ่านง่ายขึ้น จากตัวอย่างโปรแกรมข้างต้น 3 บรรทัดแรกจะเป็นหมายเหตุ

การสั่งให้โปรแกรมจบการทำงาน

โปรแกรมจะจบการทำงานเมื่อสั่งให้โปรแกรมจบการทำงานเท่านั้น ถ้าไม่ได้สั่งให้จบการทำงานเมื่อจบโปรแกรมแล้ว หน่วยประมวลผลจะทำงานคำสั่งอื่น ๆ ที่อยู่ในหน่วยความจำต่อจากโปรแกรมของไปเรื่อยๆ ในโปรแกรม DEBUG เรียกใช้คำสั่ง INT 20h เพื่อให้โปรแกรมจบการทำงาน แต่ในโปรแกรมภาษาแอสแซมบลีทั่วไปจะเรียกใช้บริการหมายเลข 4Ch ของระบบปฏิบัติการ DOS โดยจากโปรแกรมตัวอย่างใช้คำสั่งดังนี้

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

ในโปรแกรมตัวอย่างนี้ ได้เรียกใช้บริการของ DOS ในการพิมพ์ข้อความด้วย เรียกใช้บริการหมายเลข 9 โดยใช้คำสั่ง

```
mov    ah,9h
mov    dx,offset msg1
int    21h
```

สำหรับการเรียกใช้บริการของ DOS จะกล่าวถึงในเนื้อหาสัปดาห์ถัดไป

ตัวอย่าง โครงร่างของโปรแกรมภาษาแอสแซมบลี

```
dseg    segment
;       ประกาศข้อมูล
dseg    ends
sseg    segment stack
        db      100 dup (?)
sseg    ends
cseg    segment
        assume cs:cseg,ds:dseg,ss:sseg
start:
```

```
        mov     ax,dseg;ตั้งค่า DS
        mov     ds,ax
;       ตัวโปรแกรม
        mov     ax,4c00h        ;จบโปรแกรม
        int     21h
cseg    ends
        end     start
```

7.2 รูปแบบของโปรแกรมภาษาแอสแซมบลีแบบใหม่

รูปแบบของโปรแกรมภาษาแอสแซมบลีที่ใช้ในตอนต้นนั้นเป็นรูปแบบเก่า ในปัจจุบันโปรแกรมassembler ส่วนใหญ่มีรูปแบบในการประกาศเซกเมนต์ต่าง ๆ ให้ง่ายขึ้น โดยใช้คุณสมบัติของ MACRO ต่าง ๆโปรแกรมตัวอย่างแรกของเมื่อนำมาเขียนในรูปแบบใหม่จะได้เป็น

```
; This program prints the message "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
```

จะสังเกตได้ว่าโปรแกรมกระทัดรัดขึ้นมาก ข้อแตกต่างของโปรแกรมที่เขียนในรูปแบบใหม่คือชื่อของเซกเมนต์ต่าง ๆ จะถูกกำหนดให้โดยอัตโนมัติ จะสังเกตได้ว่าในส่วนของการกำหนดค่า DS ใช้ชื่อของเซกเมนต์ข้อมูลว่า @data เป็นต้น

7.3 การเรียกใช้บริการของ DOS

สามารถเรียกใช้บริการต่าง ๆ ของ DOS ได้โดยผ่านทางการขัดจังหวะหมายเลข 21h DOS ได้จัดสรรบริการ (function) ต่าง ๆ มากมายให้กับผู้เขียนโปรแกรม เมื่อเรียกใช้บริการจะต้องระบุว่าต้องการบริการใด ระบุโดยกำหนดค่าหมายเลขของบริการลงในรีจิสเตอร์ AH พร้อมทั้งข้อมูลต่าง ๆ ของการเรียกใช้บริการนั้น (พารามิเตอร์ต่าง ๆ) รูปแบบคร่าว ๆ ของการเรียกใช้บริการของ DOS เป็นดังนี้

```
mov     ah,function_number      ;(set function parameters)
int     21h
```

บริการต่าง ๆ ของ DOS ที่สำคัญ และพารามิเตอร์ของบริการต่าง ๆ มีดังต่อไปนี้

ตาราง 7.1 แสดงบริการของ DOS ที่สำคัญและพารามิเตอร์

หมายเลข	หน้าที่	พารามิเตอร์	หมายเหตุ
01h	รับค่าจากแป้นพิมพ์	Input : AH = 01h Output :AL = รหัสแอสกีของปุ่มที่กด	
02h	แสดงตัวอักษร	Input : AH = 02h DL = รหัสแอสกีของอักษรที่จะแสดง	
05h	พิมพ์ตัวอักษรทางเครื่องพิมพ์	Input : AH = 05h DL = รหัสแอสกีของอักษรที่จะพิมพ์	
07h	อ่านตัวอักษรจากแป้นพิมพ์ แต่ไม่แสดงผล (ไม่ตรวจสอบ Crtl–Break)	Input : AH = 07h Output :AL = รหัสแอสกีของอักษรที่อ่านได้	
08h	อ่านตัวอักษรจากแป้นพิมพ์ แต่ไม่แสดงผล (ตรวจสอบ Crtl–Break)	Input : AH = 07h Output :AL = รหัสแอสกีของอักษรที่อ่านได้	

หมายเลข	หน้าที่	พารามิเตอร์	หมายเหตุ
09h	พิมพ์ข้อความ	Input : AH = 09h DS:DX = ตำแหน่งของข้อความที่ต้องการพิมพ์ ข้อความจบด้วยอักษร '$'	การประกาศข้อมูลในหน่วยความจำจะอธิบายในบทถัดไป
0Ah	อ่านข้อความ	Input : AH = 0Ah DS:DX = ตำแหน่งของบัฟเฟอร์เก็บข้อมูล	รูปแบบของบัฟเฟอร์และการใช้บริการนี้จะอธิบายในบทถัดไป
4Ch	จบโปรแกรม	Input : AH = 4Ch AL = รหัสที่จะส่งคือสู่ระบบ	

7.4 ขั้นตอนการแปลโปรแกรม

ผู้เขียนโปรแกรมจะต้องแปลโปรแกรมที่เขียนขึ้นให้อยู่ในรูปแบบที่สามารถทำงานได้จริง โดยขั้นตอนการแปลโปรแกรมเป็นดังนี้

1. แปลโปรแกรมเป็นแฟ้มเป้าหมาย (object file) นามสกุล OBJ โดยใช้โปรแกรม assembler ต่าง ๆ เช่น Macro Assembler (MASM) หรือ Turbo Assembler (TASM)
2. นำมาแฟ้มเป้าหมายแฟ้มเดียวหรือหลายแฟ้มมาเชื่อมโยงเข้าด้วยกันโดยใช้โปรแกรม LINK

ตัวอย่างการแปลโปรแกรม

จากโปรแกรมตัวอย่าง สมมติว่าโปรแกรมให้เก็บในแฟ้มชื่อ EX1.ASM สามารถสั่งแปลโปรแกรมโดยใช้ Macro Assembler ได้ดังนี้

```
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
```

ถ้าโปรแกรมมีข้อผิดพลาด assembler จะแจ้งข้อผิดพลาดกลับมาให้ทราบ สามารถแก้ไขและแปลโปรแกรมใหม่ได้ เมื่อแปลโปรแกรมภาษาแอสแซมบลีเรียบร้อยแล้ว จะได้แฟ้มเป้าหมายที่มีนามสกุลเป็น OBJ เช่น จากตัวอย่างจะได้ EX1.OBJ ซึ่งจะให้โปรแกรม LINK เพื่อแปลแฟ้มเป้าหมาย (Object file) ให้เป็นโปรแกรมที่สามารถทำงานได้ ดังนี้

```
A:\>link ex1;
Microsoft (R) Segmented-Executable Linker  Version 5.13
Copyright (C) Microsoft Corp 1984-1991.  All rights reserved.
```

จะได้แฟ้มที่มี**นามสกุล EXE** ซึ่งสามารถเรียกใช้ได้จาก DOS prompt

ตัวอย่างโปรแกรม

ตัวอย่างที่ 1

โปรแกรมนี้รับการกดปุ่มจากผู้ใช้โดยใช้บริการหมายเลข 01h แล้วแสดงอักขระที่อ่านได้โดยใช้บริการของ DOS หมายเลข 02h สังเกตว่าโปรแกรมนี้ไม่มีการใช้ข้อมูลในหน่วยความจำ ดังนั้นจึงไม่ต้องประกาศเซกเมนต์ข้อมูล

```
;Example 1
.model  small
.dosseg
.stack  100h
.code
start:
        mov     ah,01h          ;read character (Function 01h)
        int     21h
        mov     dl,al           ;copy character to DL
        mov     ah,02h          ;display it (Function 02h)
        int     21h
        mov     ax,4C00h        ;Exit (Function 4Ch)
        int      21h
end  start
```

ตัวอย่างที่ 2

โปรแกรมนี้รับการกดปุ่มจากผู้ใช้โดยใช้บริการหมายเลข 01h แล้วแสดงอักขระที่มีรหัสแอสกีถัดจากอักขระที่อ่านได้ การแสดงตัวอักษรใช้บริการของ DOS หมายเลข 02h เช่นเดียวกับตัวอย่างที่ 1 โปรแกรมนี้ไม่มีการใช้ข้อมูลในหน่วยความจำจึงไม่มีการประกาศเซกเมนต์ข้อมูล โปรแกรมนี้เขียนโดยใช้รูปแบบในการเขียนแบบเก่า

```
; Example 2
sseg    segment stack
        db      100 dup (?)
sseg    ends
cseg    segment
        assume  cs:cseg,ss:sseg
start:
        mov     ah,01h          ;read character (Function 01h)
        int     21h
        mov     dl,al           ;copy to DL
        inc     dl              ;increse DL (next char.)
        mov     ah,02h          ;display it (Function 02h)
        int     21h
        mov     ax,4C00h        ;Exit
        int     21h
cseg    ends
        end     start
```

ตัวอย่างที่ 3

โปรแกรมนี้รับตัวอักษรจากผู้ใช้ จากนั้นแปลงตัวอักษรเล็กให้เป็นตัวอักษรใหญ่โดยการลบค่ารหัสแอสกีด้วย 32 แล้วแสดงอักขระนั้นกับผู้ใช้

```
.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
```

ตัวอย่างที่ 4

โปรแกรมนี้ทำงานเหมือนโปรแกรมในตัวอย่างที่ 3 แต่ไม่แสดงอักษรที่ผู้ใช้ป้อนให้ผู้ใช้เห็น โดยใช้บริการหมายเลข 08h แทนบริการหมายเลข 01h ในตัวอย่างที่ 3

```
sseg    segment stack
        db      100 dup (?)
sseg    ends
cseg    segment
          assume  cs:cseg,ss:sseg
start:
        mov     ah,08h          ; read char (Function 08h)
        int     21h
        mov     dl,al
        sub     dl,32           ; Change case
        mov     ah,02h
        int     21h
        mov     ax,4C00h        ; exit
        int     21h
cseg  ends
        end     start
```

7.5 ตารางแสดงรหัสแอสกี

ตารางที่ 7.2 แสดงตารางรหัสแอสกี (ASCII: American Standard Code for Information Interchange)

0	*NUL*	16	*DLE*	32	*SP*	48	0	64	@	80	P	96	‘	112	p
1	*SOH*	17	*DC1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*STX*	18	*DC2*	34	”	50	2	66	B	82	R	98	b	114	r
3	*ETX*	19	*DC3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*EOT*	20	*DC4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*ENQ*	21	*NAK*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ACK*	22	*SYN*	38	&	54	6	70	F	86	V	102	f	118	v
7	*BEL*	23	*ETB*	39	’	55	7	71	G	87	W	103	g	119	w
8	*BS*	24	*CAN*	40	(	56	8	72	H	88	X	104	h	120	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	105	I	121	y
10	*LF*	26	*SUB*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*VT*	27	*ESC*	43	+	59	;	75	K	91	[	107	k	123	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	108	l	124	l
13	*CR*	29	*GS*	45	–	61	=	77	M	93	]	109	m	125	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	110	n	126	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	111	o	127	

รหัสหมายเลข 0 – 31 เป็นรหัสควบคุม รหัส 32 คือช่องว่าง

สรุป

ภาษาแอสแซมบลี (Assembly Language) เป็นภาษาที่ใช้สัญญาลักษณ์ในการสื่อสาร ความหมายภาษาแอสแซมบลีมีลักษณะคำสั่ง ที่ขึ้นกับเครื่องคอมพิวเตอร์ที่ใช้งานและมีการแปลคำสั่งให้เป็นภาษาเครื่องนอกจากภาษาเครื่อง และ ภาษาแอสแซมบลีแล้ว ก็ยังมีภาษาระดับสูง เช่น Basic Cobol Fortran ซึ่งเป็นภาษาที่มีคำสั่งใกล้เคียงกับภาษาอังกฤษมากทำให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้สะดวกและรวดเร็ว แต่ว่าโปรแกรมที่เขียนด้วยภาษาระดับสูงต้องใช้เนื้อที่เก็บในหน่วยความจำเป็นจำนวนมาก อีกทั้งทำงานได้ช้ากว่าภาษาแอสแซมบลี ดังนั้นภาษาระดับสูงจึงไม่นิยมนำมาประยุกต์ใช้กับการทำงานที่ระบบการควบคุมที่มีความสำคัญมาก

คำสั่งปฏิบัติการของภาษาแอสแซมบลี แบ่งออกเป็น 4 ชนิด คือ

1. Machine instruction เป็นคำสั่งที่ทำให้เกิดการปฏิบัติการ (execution) ชุดของคำสั่งอยู่ใน assembler's instruction

2. Assembler instruction เป็นคำสั่งที่บอกแอสเซมเบลร์ให้ทำการระหว่าง Assembly source program

3. Macro instruction เป็นคำสั่งที่บอกแอสสเซมเบลร์ให้ดำเนินการกับชุดของคำสั่งที่ได้บอกไว้ก่อนแล้ว ซึ่งจากชุดของคำสั่งนี้ แอสเซมเบลร์จะผลิตชุดของคำสั่งซึ่งต่อไปจะดำเนินการเหมือนหนึ่งว่าชุดของคำสั่งนี้เป็นส่วนหนึ่งของ source program แต่เริ่มแรก

4. Pseudo instruction เป็นคำสั่งที่บอกให้แอสเซมเบลร์รู้ว่า ควรปฏิบัติการเช่นไรกับข้อมูลการ branch อย่างมีข้อแม้ แมคโคและ listing ซึ่งปกติแล้วคำสั่งเหล่านี้จะไม่ผลิตคำสั่งภาษาเครื่องให้

คำถามทบทวน

1. โปรแกรมที่ทำงานในเครื่องคอมพิวเตอร์ซึ่งใช้หน่วยประมวลผลตระกูล 80x86 นั้นจะมีการแบ่งโปรแกรมทั้งหมดเป็นเซกเมนต์ย่อย ๆ อะไรบ้าง
2. จงอธิบายขั้นตอนการแปลโปรแกรมในภาษาแอสแซมบลี
3. จงแสดงรูปแบบวิธีการเขียนโปรแกรมภาษาแอสแซมบลีแบบใหม่
4. แฟ้มเป้าหมาย (Object file) คืออะไรและมีความสัมพันธ์กันอย่างไรกับการเขียนโปรแกรมภาษาแอสแซมบลี
5. จงเขียนโปรแกรมนี้รับตัวอักษรจากผู้ใช้ จากนั้นแปลงตัวอักษรใหญ่ให้เป็นตัวอักษรเล็กออกทางหน้าจอคอมพิวเตอร์ (Display on Screen)

เอกสารอ้างอิง

ราชบัณฑิตยสถาน. (2544). *ศัพท์บัญญัติ ราชบัณฑิตยสถาน.* ค้นเมื่อ 5 มกราคม 2557,จาก :http://rirs3.royin.go.th/coinages/

รหัสแอสกี. (2557). *วิกิพีเดีย สารานุกรมเสรี.* ค้นเมื่อ 5 มกราคม 2557, จาก: http://th.wikipedia.org/wiki/

ชูชัย ธนสารตั้งเจริญ, กำธร พานิชปฐมพงษ์. *ภาษาแอสแซมบลี 80286/80386(PC).* กรุงเทพฯ :สำนักพิมพ์ซีเอ็ดยูเคชั่น บมจ, 2536.

ธีรวัฒน์ ประกอบผล. *ระบบคอมพิวเตอร์และภาษาแอสแซมบลี.* กรุงเทพฯ :สำนักพิมพ์ส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2537.