แผนบริหารการสอนประจำบทที่ 7

หัวข้อเนื้อหา

- รูปแบบของโปรแกรมภาษาแอสเซมบลีแบบเก่า
- รูปแบบของโปรแกรมภาษาแอสเซมบลีแบบใหม่
- ขั้นตอนการแปลโปรแกรม
- การเรียกใช้บริการของ DOS
- ตารางรหัสแอสกี

วัตถุประสงค์เชิงพฤติกรรม

- เข้าใจรูปแบบของการเขียนโปรแกรมภาษาแอสเซมบลี การประกาศเซกเมนต์แบบต่างๆ เช่น เซกเมนต์ cseg dseg และ sseg เป็นต้น
- สามารถและเข้าใจการประกาศข้อมูล การใส่หมายเหตุ และการสั่งให้โปรแกรมจบการทำงาน
- สามารถและเข้าใจการเรียกใช้บริการของ DOS และอ่านตารางรหัสแอสกีได้

วิธีสอนและกิจกรรมการเรียนการสอน

- บรรยาย
- สืบเสาะหาความรู้
- ค้นคว้าเพิ่มเติม
- ตอบคำถาม

สื่อการเรียนการสอน

- สื่ออิเล็กทรอนิกส์
- เอกสารอ้างอิงประกอบการค้นคว้า

การวัดผลและประเมินผล

ใช้วิธีการสังเกตและจดบันทึกไว้เป็นระยะ

- สังเกตจากงานที่กำหนดให้ไปทำมาส่ง
- สังเกตจากการตอบคำถาม
- สังเกตจากการนำความรู้ไปใช้

การประเมินผล

วิธีตรวจผลงานต่างๆ ที่ให้ทำ

- ตรวจผลงานภาคปฏิบัติ
- ตรวจรายงาน
- ตรวจแบบฝึกหัด

ใช้วิธีการออกข้อสอบข้อเขียน

บทที่ 7 โปรแกรมภาษาแอสเซมบลีเบื้องต้น (Assembly Language Progamming)

ในบทนี้เราจะศึกษาการเขียนโปรแกรมภาษาแอสเซมบลีแบบเต็มรูปแบบ นั่นคือเราจะเขียนโปรแกรมภาษาแอสเซมบลีที่เป็นโปรแกรมที่ทำงานได้จริง มีการกำหนดรูปแบบต่างๆ ครบถ้วน โปรแกรมภาษาแอสเซมบลีที่เราจะเขียนต่อไปนี้ไม่ได้ทำงานบนโปรแกรม DEBUG เท่านั้น เราจะต้องใช้ assembler แปลโปรแกรมที่เราเขียนขึ้นให้อยู่ในรูปที่คอมพิวเตอร์สามารถนำไปประมวลผลได้เสียก่อน

รูปแบบของโปรแกรมภาษาแอสเซมบลี

โปรแกรมที่ทำงานในเครื่องคอมพิวเตอร์ซึ่งใช้หน่วยประมวลผลตระกูล 80x86 นั้นจะมีการแบ่งโปรแกรมทั้งหมดเป็นเซกเมนต์ย่อย ๆ เช่น Code segment Data segment หรือ Stack segment ดังนั้นในโปรแกรมภาษาแอสเซมบลีที่เราเขียนจะประกอบไปด้วยเซกเมนต์ต่าง ๆ เช่นเดียวกัน ภายในเซกเมนต์ต่าง ๆ ที่เราประกาศเราจะระบุข้อมูลและโปรแกรมที่จะอยู่ในเซกเมนต์นั้น

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 เราจะสังเกตได้ว่าโปรแกรมได้ประกาศเซกเมนต์ทั้งหมด 3 เซกเมนต์ คือ cseg dseg และ sseg เซกเมนต์ดังกล่าวนี้ถูกประกาศด้วย**คำสั่งเทียม segment** การที่เราเรียกคำสั่ง segment ว่าคำสั่งเทียม เพราะคำสั่งนี้เป็นคำสั่งที่ผู้เขียนโปรแกรมระบุให้โปรแกรม assembler แปลโปรแกรมตามลักษณะที่กำหนด โดยจะไม่มีคำสั่งภาษาเครื่องถูกสร้างจากคำสั่งกลุ่มนี้ ตัวอย่างอื่น ๆ ของคำสั่งเทียมคือ db assume และ org เป็นต้น

การประกาศเซกเมนต์

การประกาศเซกเมนต์ในโปรแกรมภาษาเอสเซมบลี เราใช้คู่คำสั่งเทียม segment และ ends โดยมีลักษณะการประกาศดังนี้.

```
segment_name        segment
        ...
segment_name        ends
```

จากตัวอย่างเราได้ประกาศเซกเมนต์ cseg dseg และ sseg **คำสั่งเทียม stack** ระบุให้ระบบใช้เซกเมนต์ sseg เป็นแสต็กของโปรแกรม **คำสั่งเทียม assume** เป็นการระบุให้ assembler ได้ทราบว่าเซกเมนต์ที่เราประกาศนั้นจะให้ระบบพิจารณาว่าทำหน้าที่อะไรและมีเซกเมนต์รีจิสเตอร์ใดเป็นตัวเก็บค่าเซกเมนต์ จากตัวอย่างเราประกาศให้ assembler ทราบว่าเซกเมนต์ cseg จะชี้โดยรีจิสเตอร์ CS เซกมนต์ dseg จะชี้โดยรีจิสเตอร์ DS และเซกเมนต์ sseg จะชี้โดยรีจิสเตอร์ SS คำสั่งเทียม assume นี้จะเป็นการบอก assembler ให้พิจารณาตามที่ระบุเท่านั้น ไม่ได้เป็นการสั่งให้ assembler กำหนดค่าต่าง ๆ ให้โดยอัตโนมัติ สังเกตได้จากในตอนต้นของโปรแกรมเรามีชุดคำสั่งเพื่อปรับค่าของรีจิสเตอร์ DS ดังนี้

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

ชุดคำสั่งนี้จะปรับค่าของรีจิสเตอร์ DS ให้ชี้ไปที่ dseg สำหรับรีจิสเตอร์ SS ระบบจะปรับค่าให้ชี้ไปที่เซกเมนต์ที่เราระบุไว้โดยคำสั่งเทียม stack ส่วนกรณีของรีจิสเตอร์ CS นั้นระบบจะตั้งค่าให้ตรงกับเซกเมนต์ที่เริ่มต้นโปรแกรม

โปรแกรมภาษาแอสเซมบลีจะประกอบไปด้วยการประกาศเซกเมนต์ต่าง ๆ และจะสิ้นสุดโปรแกรมที่ **คำสั่งเทียม end** หลังคำสั่งเทียม end เราจะระบุจุดเริ่มต้นของโปรแกรม ในโปรแกรมตัวอย่างเราระบุจุดเริ่มต้นของโปรแกรมที่เลเบล start ที่เราประกาศไว้ที่ตอนต้นของโปรแกรม การประกาศเลเบลเราสามารถทำได้ดังนี้

label_name:

ระบบจะจดจำตำแหน่งของเลเบลที่เราประกาศไว้และจะนำแอดเดรสของเลเบลไปแทนที่ให้โดยอัตโนมัติ. การที่โปรแกรม assembler จัดการเรื่องเกี่ยวกับเลเบลในโปรแกรมภาษาแอสเซมบลีนั้น นับเป็นการเพิ่มความสะดวกให้กับผู้เขียนโปรแกรมเป็นอย่างมาก

การประกาศข้อมูล

ภายในเซกเมนต์ข้อมูลเราสามารถประกาศข้อมูลต่าง ๆ ได้ จากโปรแกรมตัวอย่างเราประกาศข้อมูลเป็นข้อความที่จะให้โปรแกรมพิมพ์ออกมา เราจะศึกษารูปแบบการประกาศข้อมูลในบทต่อไป

การใส่หมายเหตุ

หลังเครื่องหมาย ‘;’ assembler จะตีความว่าเป็นหมายเหตุ การใส่หมายเหตุจะช่วยทำให้โปรแกรมอ่านง่ายขึ้น จากตัวอย่างโปรแกรมข้างต้น 3 บรรทัดแรกจะเป็นหมายเหตุ

การสั่งให้โปรแกรมจบการทำงาน

โปรแกรมจะจบการทำงานเมื่อเราสั่งให้โปรแกรมจบการทำงานเท่านั้น ถ้าเราไม่ได้สั่งให้จบการทำงานเมื่อจบโปรแกรมแล้ว หน่วยประมวลผลจะทำงานคำสั่งอื่น ๆ ที่อยู่ในหน่วยความจำต่อจากโปรแกรมของเราไปเรื่อยๆ ในโปรแกรม DEBUG เราเรียกใช้คำสั่ง INT 20h เพื่อให้โปรแกรมจบการทำงาน แต่ในโปรแกรมภาษาแอสเซมบลีทั่วไปเราจะเรียกใช้บริการหมายเลข 4Ch ของระบบปฏิบัติการ DOS โดยจากโปรแกรมตัวอย่างเราใช้คำสั่งดังนี้

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

ในโปรแกรมตัวอย่างนี้ เราได้เรียกใช้บริการของ DOS ในการพิมพ์ข้อความด้วย เราเรียกใช้บริการหมายเลข 9 โดยใช้คำสั่ง

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

สำหรับการเรียกใช้บริการของ DOS เราจะกล่าวถึงในหัวข้อถัดไป.

ตัวอย่างโครงร่างของโปรแกรมภาษาแอสเซมบลี

```
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
```

รูปแบบของโปรแกรมภาษาแอสเซมบลีแบบใหม่

รูปแบบของโปรแกรมภาษาแอสเซมบลีที่เราใช้ในตอนต้นนั้นเป็นรูปแบบเก่า ในปัจจุบันโปรแกรม assembler ส่วนใหญ่มีรูปแบบในการประกาศเซกเมนต์ต่าง ๆ ให้ง่ายขึ้น โดยใช้คุณสมบัติของ MACRO ต่าง ๆ โปรแกรมตัวอย่างแรกของเราเมื่อนำมาเขียนในรูปแบบใหม่จะได้เป็น

```
;
; 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 เราจะสังเกตได้ว่าในส่วนของการกำหนดค่า DS เราใช้ชื่อของเซกเมนต์ข้อมูลว่า @data เป็นต้น

การเรียกใช้บริการของ DOS

เราสามารถเรียกใช้บริการต่าง ๆ ของ DOS ได้โดยผ่านทางการขัดจังหวะหมายเลข 21h DOS ได้จัดสรรบริการ (function) ต่าง ๆ มากมายให้กับผู้เขียนโปรแกรม. เมื่อเราเรียกใช้บริการเราจะต้องระบุว่าต้องการบริการใด. เราระบุโดยกำหนดค่าหมายเลขของบริการลงในรีจิสเตอร์ 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 บริการของ 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	
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	

ขั้นตอนการแปลโปรแกรม

เราจะต้องแปลโปรแกรมที่เราเขียนขึ้นให้อยู่ในรูปแบบที่สามารถทำงานได้ โดยขั้นตอนการแปลโปรแกรมเป็นดังนี้.

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

ตัวอย่างการแปลโปรแกรม

จากโปรแกรมตัวอย่าง สมมติว่าเราเก็บในแฟ้มชื่อ EX1.ASM เราสามารถสั่งแปลโปรแกรมโดยใช้ 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 เราสามารถแก้ไขและแปลโปรแกรมใหม่ได้ เมื่อเราแปลโปรแกรมภาษาแอสเซมบลีเรียบร้อยแล้ว เราจะได้แฟ้มเป้าหมายที่มีนามสกุลเป็น OBJ เช่นจากตัวอย่างเราจะได้ EX1.OBJ เราจะให้โปรแกรม 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.
```

เราจะได้แฟ้มที่มีนามสกุล EXE ซึ่งสามารถเรียกใช้ได้จาก DOS prompt

ตัวอย่างโปรแกรม

ตัวอย่างที่ 1

โปรแกรมนี้รับการกดปุ่มจากผู้ใช้โดยใช้บริการหมายเลข 01h แล้วแสดงอักขระที่อ่านได้โดยใช้บริการของ DOS หมายเลข 02h สังเกตว่าโปรแกรมนี้ไม่มีการใช้ข้อมูลในหน่วยความจำ ดังนั้นจึงไม่ต้องประกาศเซกเมนต์ข้อมูล

```
;Ex1
.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

```
;Ex2
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
```

หมายเหตุ ตารางรหัสแอสกี

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	*EN*	21	*NA*	37	%	53	5	69	E	85	U	101	e	117	u

	Q		*K*									1		7	
6	*AC K*	22	*SY N*	38	&	54	6	70	F	86	V	10 2	f	11 8	v
7	*BE L*	23	*ET B*	39	’	55	7	71	G	87	W	10 3	g	11 9	w
8	*BS*	24	*CA N*	40	(	56	8	72	H	88	X	10 4	h	12 0	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	10 5	I	12 1	y
10	*LF*	26	*SU B*	42	*	58	:	74	J	90	Z	10 6	j	12 2	z
11	*VT*	27	*ES C*	43	+	59	;	75	K	91	[	10 7	k	12 3	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	10 8	l	12 4	|
13	*CR*	29	*GS*	45	-	61	=	77	M	93	]	10 9	m	12 5	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	11 0	n	12 6	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	11 1	o	12 7	

รหัสหมายเลข 0 - 31 เป็นรหัสควบคุม รหัส 32 คือช่องว่าง

สรุป

ภาษาแอสเซมบลี (Assembly Language) เป็นภาษาที่ใช้สัญญาลักษณ์ในการสื่อสารความหมาย ภาษาแอสเซมบลีมีลักษณะคำสั่ง ที่ขึ้นกับเครื่องคอมพิวเตอร์ที่ใช้งานและมีการแปลคำสั่งให้เป็นภาษาเครื่อง นอกจากภาษาเครื่อง และ ภาษาแอสเซมบลีแล้ว ก็ยังมีภาษาระดับสูง เช่น Basic Cobol Fortran ซึ่งเป็นภาษาที่มีคำสั่งใกล้เคียงกับภาษาอังกฤษมากทำให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้สะดวกและรวดเร็ว แต่ว่าโปรแกรมที่เขียนด้วยภาษาระดับสูงต้องใช้เนื้อที่เก็บในหน่วยความจำเป็นจำนวนมาก อีกทั้งทำงานได้ช้ากว่าภาษาแอสเซมบลี ดังนั้นภาษาระดับสูงจึงไม่นิยมนำมาประยุกต์ใช้กับการทำงานที่ระบบการควบคุมที่มีความสำคัญมาก

ภาษาแอสเซมบลี เหมาะกับโปรแกรมที่ใช้เนื้อที่ในหน่วยความจำไม่มากนัก ทั้งทำงานได้รวดเร็ว และในการควบคุมการทำงานของเครื่องคอมพิวเตอร์ได้โดยตรง

คำสั่งปฏิบัติการของภาษาแอสเซมบลี แบ่งออกเป็น 4 ชนิดคือ

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

2. Assembler instruction เป็นคำสั่งที่บอกแอสเซมเบลร์ให้ทำการระหว่างการแอสเซมบลี 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. จงอธิบายขั้นตอนการแปลโปรแกรมในภาษาแอสเซมบลี

3. จงแสดงรูปแบบวิธีการเขียนโปรแกรมภาษาแอสเซมบลีแบบใหม่

4. แฟ้มเป้าหมาย (Object file) คืออะไรและมีความสัมพันธ์กันอย่างไรกับการเขียนโปรแกรมภาษาแอสเซมบลี

5. จงเขียนโปรแกรมนี้รับตัวอักษรจากผู้ใช้ จากนั้นแปลงตัวอักษรใหญ่ให้เป็นตัวอักษรเล็กออกทางหน้าจอคอมพิวเตอร์ (Display on Screen)