แผนบริหารการสอนประจำบทที่ 5

หัวข้อเนื้อหา

- คำสั่งในการโยนย้ายข้อมูล คำสั่ง MOV
- เครื่องมือในการทดลองการโปรแกรมภาษาแอสเซมบลี
- คำสั่งในการโยนย้ายข้อมูล คำสั่งทั่วไป

วัตถุประสงค์เชิงพฤติกรรม

- เข้าใจความหมายและขั้นตอนการทำงานของคำสั่งในการโยนย้ายข้อมูล
- รู้และเข้าใจ สามารถใช้งานโปรแกรม DEBUG ภาษาแอสเซมบลีได้
- รู้และเข้าใจการใช้งานคำสั่งทั่วไปและการ DEBUG เพื่อตรวจสอบความถูกต้องของคำสั่งโปรแกรมภาษาแอสเซมบลีได้

วิธีสอนและกิจกรรมการเรียนการสอน

- บรรยาย
- สืบเสาะหาความรู้
- ค้นคว้าเพิ่มเติม
- ตอบคำถาม

สื่อการเรียนการสอน

- สื่ออิเล็กทรอนิกส์
- เอกสารอ้างอิงประกอบการค้นคว้า

การวัดผลและประเมินผล

ใช้วิธีการสังเกตและจดบันทึกไว้เป็นระยะ

- สังเกตจากงานที่กำหนดให้ไปทำมาส่ง
- สังเกตจากการตอบคำถาม
- สังเกตจากการนำความรู้ไปใช้

การประเมินผล

วิธีตรวจผลงานต่างๆ ที่ให้ทำ

- ตรวจผลงานภาคปฏิบัติ
- ตรวจรายงาน
- ตรวจแบบฝึกหัด

ใช้วิธีการออกข้อสอบข้อเขียน

บทที่ 5 คำสั่งโอนย้ายข้อมูล (Command Transfer)

ในบทนี้ เราจะเริ่มศึกษาเกี่ยวกับคำสั่งภาษาแอสเซมบลีเบื้องต้น โดยจะเป็นคำสั่งเกี่ยวกับการโอนย้ายข้อมูล ทั้งจากรีจิสเตอร์และหน่วยความจำ ในตอนท้ายของบทนี้จะเป็นการแนะนำโปรแกรม DEBUG ซึ่งจะเป็นเครื่องมือที่เราจะใช้ทดลองการโปรแกรมภาษาแอสเซมบลีเบื้องต้น

คำสั่ง MOV

คำสั่ง MOV เป็นคำสั่งที่ใช้สำหรับการคัดลอกข้อมูลจากแหล่งข้อมูลต้นทางไปยังแหล่งข้อมูลปลายทาง โดยมีรูปแบบดังนี้

```
MOV     reg, reg
MOV     reg, mem
MOV     mem, reg
MOV     reg, imm
MOV     mem, imm
```

reg : register
mem : memory
imm : immediate (ค่าคงที่)

ในการคัดลอกข้อมูลจะกระทำจากโอเปอร์แรนด์ (operand) ตัวหลังไปยังโอเปอร์แรนด์ตัวหน้า สังเกตว่า เราไม่สามารถคัดลอกข้อมูลจากหน่วยความจำไปยังหน่วยความจำได้

ข้อจำกัดของคำสั่ง MOV

- โอเปอร์แรนด์ทั้งสองตัวต้องมีขนาดเท่ากัน
- ไม่สามารถคัดลอกค่าคงที่ (immediate) ไปยังเซกเมนต์รีจิสเตอร์ได้โดยตรง ต้องกระทำผ่านทางรีจิสเตอร์อื่น ๆ เช่น AX เป็นต้น
- ในการคัดลอกค่าคงที่ไปยังหน่วยความจำจะต้องระบุขนาดของหน่วยความจำด้วย

ตัวอย่าง

```
MOV     AX,100h
MOV     BX,AX
MOV     DX,BX
```

กำหนดค่ารีจิสเตอร์ AX ให้เป็น 100h จากนั้นคัดลอกค่าจาก AX ไปยัง BX และคัดลอกจาก BX ไปยัง DX ตามลำดับ

```
MOV     AX,1234h
MOV     DX,5678h
MOV     AL,DL
MOV     BH,DH
```

กำหนดค่า 1234h ให้กับ AX และ 5678h ให้กับ DX จากนั้น คัดลอกค่าในรีจิสเตอร์ DL (มีค่าเท่ากับ 78h เพราะอะไร?) ไปให้กับรีจิสเตอร์ AL และคัดลอกค่าจาก DH (มีค่าเท่ากับ 56h เพราะอะไร?) ไปยังรีจิสเตอร์ DH

MOV	AX,1000h	กำหนดค่า 1000h ให้กับ AX จากนั้นคัดลอกข้อมูลจาก
MOV	[100h],AX	AX ไปยังหน่วยความจำตำแหน่ง offset ที่ 100h และ
MOV	BX,[100h]	คัดลอกข้อมูลจากหน่วยความจำตำแหน่งนั้นมายัง BX
MOV	BYTE PTR [200h],10h	กำหนดค่า 10h ไปยังตำแหน่งในหน่วยความจำที่ offset
MOV	WORD PTR [300h],10h	200h โดยโอนย้ายข้อมูลแบบขนาด 1 Byte และ กำหนดค่า 10h ที่มีขนาด 1 word (0010h) ไปยัง ตำแหน่งในหน่วยความจำที่ offset 300h
MOV	AX,2300h	กำหนดค่า 2300h ให้กับรีจิสเตอร์ DS โดยต้องกำหนด
MOV	DS,AX	ผ่านรีจิสเตอร์ขนาด 16 บิตตัวอื่น (ในที่นี้คือ AX)

การโอนย้ายข้อมูลระหว่างรีจิสเตอร์กับรีจิสเตอร์

การโอนย้ายข้อมูลระหว่างรีจิสเตอร์สามารถทำได้ถ้าขนาดของรีจิสเตอร์ทั้งคู่เท่ากัน แต่ยังมีเซกเมนต์รีจิสเตอร์บางตัวซึ่งไม่ควรเข้าไปกำหนดค่าโดยตรง เช่น CS หรือ SS

คู่รีจิสเตอร์ 16 บิต กับ 8 บิต

ในการใช้งานรีจิสเตอร์ทั่วไปเราจะต้องคำนึงถึงเรื่องของคู่รีจิสเตอร์ 16 บิต กับ 8 บิตด้วย จากที่เราได้ทราบมาแล้วว่ารีจิสเตอร์ 8 บิตที่เราใช้ได้นั้น เป็นส่วนหนึ่งของรีจิสเตอร์ทั่วไปขนาด 16 บิต เช่น รีจิสเตอร์ AH เป็นไบต์สูงรีจิสเตอร์ AX (บิตที่ 8-15) เป็นต้น ดังนั้นถ้าเรากำหนดค่าให้กับรีจิสเตอร์ 8 บิตก็จะมีการเปลี่ยนแปลงกระทบถึงรีจิสเตอร์ 16 บิตที่รีจิสเตอร์นั้นประกอบอยู่ด้วย และในทางกลับกัน การเปลี่ยนแปลงค่าในรีจิสเตอร์ 16 บิตก็มีผลกระทบมาถึงรีจิสเตอร์ 8 บิตด้วย

ตัวอย่าง

คำสั่ง		ค่าในริจิสเตอร์หลังการทำงานของคำสั่ง		
MOV	AX,1000h	AX = 1000h	AH = 10h	AX = 00h
MOV	AL,3Ah	AX = 103Ah	AH = 10h	AX = 3Ah
MOV	AH,AL	AX = 3A3Ah	AH = 3Ah	AX = 3Ah
MOV	AX,234h	AX = 234h	AH = 02h	AX = 34h

การโอนย้ายข้อมูลกับหน่วยความจำ

โดยปกติการโอนย้ายข้อมูลกับหน่วยความจำนั้น เราจะระบุเฉพาะออฟเซ็ตของหน่วยความจำที่เราต้องการจะโอนย้ายข้อมูลด้วย ออฟเซ็ตนั้นจะถูกนำมาประกอบกับเซกเมนต์รีจิสเตอร์ DS เพื่อเป็นตำแหน่งในหน่วยความจำที่แท้จริงที่จะอ่านเขียนข้อมูลด้วย

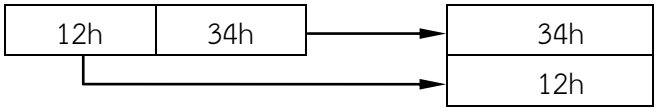
ตัวอย่าง

โปรแกรม	ตำแหน่ง	ค่าในหน่วยความจำหลังการทำงานของคำสั่งที่						
MOV AX,1234h	DS:	1,2	3	4	5	6	7	8,9
MOV BX,6789h	100h	?	12h	12h	12h	12h	12h	12h
MOV [100h],AH	101h	?	?	89h	89h	89h	89h	89h
MOV [101h],BL	102h	?	?	?	89h	89h	89h	89h
MOV [102h],BX	103h	?	?	?	67h	67h	67h	67h
MOV [104h],AX	104h	?	?	?	?	34h	34h	34h
MOV [105h],BH	105h	?	?	?	?	12h	67h	67h
MOV AX,[104h]	106h	?	?	?	?	?	?	?
MOV BL,[103h]								

หลังการทำงาน รีจิสเตอร์ AX มีค่าเท่ากับ 6734h และ BX มีค่าเท่ากับ 3467h

ข้อสังเกตในการจัดเรียงไบต์เมื่อเก็บข้อมูลในหน่วยความจำ

สังเกตว่าในการเก็บค่าในหน่วยความจำเมื่อเราเก็บค่าเป็น 16 บิต การเรียงไบต์ในหน่วยความจำจะเก็บค่าในไบต์ที่มีนัยสำคัญสูงไว้ในไบต์ที่มีแอดเดรสสูงกว่า และไบต์ที่มีนัยสำคัญต่ำไว้ในแอดเดรสที่มีแอดเดรสต่ำกว่า ดังรูป



รูปที่ 5.1 แสดงการเรียงไบต์ข้อมูลในหน่วยความจำ

ลักษณะของการเก็บข้อมูลโดยเรียงไบต์ที่มีนัยสำคัญต่ำไว้ที่แอดเดรสต่ำและไบต์ที่มีนัยสำคัญสูงไว้ที่แอดเดรสสูงเราเรียกว่าเป็นการเรียงแบบ ***c***

ในการกำหนดออฟเซ็ตของหน่วยความจำที่จะอ่านและเขียนข้อมูลนั้น จากตัวอย่างข้างต้นเราระบุโดยใส่หมายเลขออฟเซ็ตโดยตรง แต่เราสามารถระบุออฟเซ็ตโดยผ่านทางค่าในรีจิสเตอร์ BX (base register) ได้อีกทางหนึ่ง

ตัวอย่าง

```
MOV   BX,100h
MOV   AX,[BX]
MOV   BX,102h
MOV   [BX],DX
```

การกำหนดค่าคงที่ให้กับหน่วยความจำ

การกำหนดค่าคงที่ให้กับหน่วยความจำสามารถกระทำได้ แต่จะต้องมีการระบุขนาดของข้อมูลที่จะคัดลอกสู่หน่วยความจำ เพื่อที่จะป้องกันการสับสนดังตัวอย่างเช่น คำสั่ง MOV [100h],10h จากคำสั่งดังกล่าวแอสเซมเบลอร์จะไม่สามารถทราบได้เลยว่าการคัดลอกข้อมูลจะเป็นในลักษณะของ 8 บิต หรือ 16 บิต

ในการระบุขนาดของการคัดลอกข้อมูลเราจะใช้ keyword ว่า BYTE PTR (Byte Pointer) หรือ WORD PTR (Word Pointer) สำหรับระบุว่าการอ้างถึงหน่วยความจำในตำแหน่งนั้นเป็นแบบไบต์ หรือ เวิร์ด

ตัวอย่าง

```
MOV    BYTE PTR [100h],10h
MOV    WORD PTR [102h],10h
MOV    BX,104h
MOV    WORD PTR [BX],2345h
```

เครื่องมือในการทดลองการโปรแกรมภาษาแอสเซมบลี : โปรแกรม DEBUG

โปรแกรม DEBUG เป็นโปรแกรมสารพัดประโยชน์สำหรับการทดลองเกี่ยวกับการเขียนโปรแกรมภาษาแอสเซมบลี โปรแกรมนี้มีอยู่ทั้งในระบบปฏิบัติการ DOS และ Windows

เมื่อเราเรียกโปรแกรม DEBUG โดยพิมพ์ DEBUG ที่ DOS prompt เราจะเป็นเครื่องหมายเตรียมพร้อมของโปรแกรม DEBUG เป็นเครื่องหมายขีด

```
-
```

เราสามารถจะพิมพ์คำสั่งต่าง ๆ ลงไปได้ที่ prompt นี้

คำสั่งทั่วไป

คำสั่งแสดงความช่วยเหลือ : ?

เราสามารถสั่งให้โปรแกรม DEBUG แสดงรายการคำสั่งต่าง ๆ ได้โดยใช้คำสั่ง ?

คำสั่งจัดการกับรีจิสเตอร์ : R (register)

คำสั่ง R คือคำสั่งให้โปรแกรม DEBUG แสดงค่าในรีจิสเตอร์รวมถึงกำหนดค่าให้กับรีจิสเตอร์ต่าง ๆ ด้วย ถ้าเราใช้คำสั่ง R โดยไม่ระบุชื่อรีจิสเตอร์ โปรแกรม DEBUG จะแสดงค่าในรีจิสเตอร์ออกมาให้

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=12AF  ES=12AF  SS=12AF  CS=12AF  IP=0100   NV UP EI PL NZ NA PO NC
12AF:0100 5F            POP     DI
-
```

โปรแกรม DEBUG จะแสดงทั้งค่าในรีจิสเตอร์ คำสั่งถัดไปที่จะทำงาน รวมถึงแฟล็กต่าง ๆ ด้วย เราสามารถระบุชื่อ รีจิสเตอร์ต่อท้ายคำสั่ง R เพื่อกำหนดค่าให้กับรีจิสเตอร์นั้น ดังตัวอย่าง

```
-RCX
CX 0000
:100
-R
AX=0000  BX=0000  CX=0100  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=12AF  ES=12AF  SS=12AF  CS=12AF  IP=0100   NV UP EI PL NZ NA PO NC
12AF:0100 5F            POP     DI
-
```

คำสั่งแสดงค่าในหน่วยความจำ : D (dump)

เราสามารถสั่งให้โปรแกรม DEBUG แสดงค่าข้อมูลในหน่วยความจำได้ โดยใช้คำสั่ง D (dump) เราสามารถระบุตำแหน่งของหน่วยความจำที่เริ่มต้นแสดงข้อมูลได้ ถ้าเราไม่กำหนดโปรแกรม DEBUG จะแสดงค่าในหน่วยความจำถัดจากตำแหน่งเก่าที่แสดงไว้

```
-D100
12AF:0100   C7 06 16 00 6E 30 26 C7-06 18 00 C7 40 26 8B 06  ....n0&.....@&..
12AF:0110   08 00 26 03 06 0C 00 26-3B 06 06 00 34 00 9E 12   ..&....&;...4...
                    . . .
12AF:0170   C2 02 00 90 C8 04 00 00-57 56 8B 7E 0A 57 E8 57   ........WV.~.W.W
-
```

การทดลองโปรแกรมภาษาแอสเซมบลีใน DEBUG

เราสามารถป้อนโปรแกรมภาษาแอสเซมบลีเบื้องต้นได้ในโปรแกรม DEBUG และสามารถสั่งให้โปรแกรมนั้นทำงานเพื่อสังเกตผลการทำงานได้ เรายังสามารถถึงสั่งให้โปรแกรมทำงานทีละบรรทัดได้ด้วย

คำสั่งสำหรับแปลโปรแกรมภาษาแอสเซมบลี : A (assemble)

เราใช้คำสั่ง A (assemble) เพื่อสั่งให้โปรแกรม DEBUG รับคำสั่งภาษาแอสเซมบลีแล้วแปลคำสั่งนั้นเป็นภาษาเครื่องเก็บไว้ในหน่วยความจำได้ ในการสั่งคำสั่ง assemble เราสามารถระบุตำแหน่งที่โปรแกรมภาษาเครื่องที่แปลแล้วจะถูกเขียนลงไปได้ ถ้าเราไม่ระบุตำแหน่งลงไปโปรแกรมที่เขียนจะถูกเก็บไว้ที่ตำแหน่งถัดจากการแปลครั้งสุดท้าย ในการป้อนโปรแกรมเราสามารถป้อนโปรแกรมต่อเนื่องไปได้เรื่อย ๆ เมื่อป้อนเสร็จแล้วให้กดปุ่ม Enter ว่างไปหนึ่งบรรทัดโปรแกรม DEBUG จะแสดง prompt เพื่อรับคำสั่งใหม่ต่อไป

```
-A100
12AF:0100 mov ax,10
12AF:0103 mov bx,20
12AF:0106 mov [200],ax
12AF:0109 mov [202],bx
12AF:010D mov bx,204
```

```
12AF:0110 mov cx,1234
12AF:0113 mov [bx],cx
12AF:0115 int 20
12AF:0117
-
```

ข้อสังเกต ตัวเลขต่าง ๆ ในโปรแกรม DEBUG จะถือว่าเป็นเลขฐานสิบหกทั้งหมด

ถ้าเราป้อนโปรแกรมผิดโปรแกรม DEBUG จะรายงานว่าโปรแกรมผิดให้เราทราบและป้อนเข้าไปใหม่

คำสั่งสำหรับแสดงโปรแกรมภาษาแอสเซมบลีจากภาษาเครื่อง : U (unassemble)

เราสามารถสั่งให้โปรแกรม DEBUG แสดงคำสั่งภาษาเครื่องที่อยู่ในหน่วยความจำเป็นรหัสคำสั่งภาษาแอสเซมบลีได้โดยใช้คำสั่ง unassemble เราสามารถระบุตำแหน่งที่จะเริ่มแสดงได้

```
-U100
12AF:0100 B81000      MOV    AX,0010
12AF:0103 BB2000      MOV    BX,0020
12AF:0106 A30002      MOV    [0200],AX
12AF:0109 891E0202    MOV    [0202],BX
12AF:010D BB0402      MOV    BX,0204
12AF:0110 B93412      MOV    CX,1234
12AF:0113 890F        MOV    [BX],CX
12AF:0115 CD20        INT    20
12AF:0117 26          ES:
12AF:0118 3B060600    CMP    AX,[0006]
12AF:011C 3400        XOR    AL,00
12AF:011E 9E          SAHF
12AF:011F 12FE        ADC    BH,DH
-
```

โปรแกรม DEBUG จะแสดงทั้งรหัสภาษาเครื่องและรหัสนีโมนิค (รหัสภาษา assembly) ด้วย

การสั่งให้โปรแกรมทำงาน

เราสามารถสั่งให้โปรแกรมทำงานได้ทั้งสิ้น 3 รูปแบบโดยโปรแกรมจะเริ่มทำงานที่ตำแหน่ง CS:IP เท่านั้น

คำสั่งเริ่มทำงาน : G (go)

เมื่อเราสั่งให้โปรแกรมทำงานโดยใช้คำสั่ง g โปรแกรมจะเริ่มทำงานทันที และโปรแกรมจะทำงานจนกระทั่งจบ

```
-G
Program terminated normally
-
```

ข้อสังเกต โปรแกรมตัวอย่างจะมีคำสั่ง INT 20h ระบุอยู่ตอนท้าย คำสั่งนี้เป็นคำสั่งเรียกใช้บริการของระบบเพื่อที่จะจบโปรแกรม ถ้าไม่มีคำสั่งนี้ปิดท้าย โปรแกรมจะทำงานตามคำสั่งต่อไปเรื่อย ๆ จนกว่าจะพบคำสั่งที่สั่งให้โปรแกรมหยุดการทำงาน ดังนั้นเวลาเราทดลงโปรแกรมใน DBUG เราควรจะใส่คำสั่ง INT 20h ปิดท้ายไว้ทุกครั้ง

คำสั่งตามรอยการทำงาน (คำสั่งให้ทำงานทีละบรรทัดแบบที่หนึ่ง) : T (trace)

ถ้าเราสั่งคำสั่งนี้โปรแกรมจะทำงานไปหนึ่งคำสั่งแล้วจะกลับมาที่โปรแกรม DEBUG และแสดงค่าของรีจิสเตอร์ต่าง ๆ รวมถึงผลจากคำสั่งนั้นทันที ทำให้เราสามารถตรวจสอบสถานะของโปรแกรมได้ตลอดขั้นตอนการทำงาน

```
-T
AX=0010  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=12AF  ES=12AF  SS=12AF  CS=12AF  IP=0103   NV UP EI PL NZ NA PO NC
12AF:0103 BB2000        MOV     BX,0020
-T
AX=0010  BX=0020  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=12AF  ES=12AF  SS=12AF  CS=12AF  IP=0106   NV UP EI PL NZ NA PO NC
12AF:0106 A30002        MOV     [0200],AX                       DS:0200=0010
```

ในการทำงานโดยคำสั่ง T โปรแกรมที่ทำงานจะหยุดการทำงานกลับมาที่โปรแกรม DEBUG ภายหลังการทำงานของทุก ๆ คำสั่ง รวมถึงในกรณีที่เราเรียกโปรแกรมย่อย หรือเรียกใช้บริการของระบบด้วย เราต้องการให้โปรแกรมทำงานไปหยุดที่บรรทัดถัดไปเลยโดยไม่ต้องคำนึงถึงการเรียกโปรแกรมย่อยต่าง ๆ เราสามารถใช้คำสั่ง P (proceed) แทนคำสั่ง trace ได้

คำสั่งทำงานต่อจนถึงบรรทัดถัดไป (คำสั่งให้ทำงานทีละบรรทัดแบบที่สอง) : P (proceed)

ผลโดยทั่ว ๆ ไปของคำสั่งนี้เหมือนกับการใช้คำสั่ง T แต่การใช้คำสั่งนี้จะสะดวกกว่าเมื่อโปรแกรมที่ทำงานมีการเรียกโปรแกรมย่อย หรือมีการเรียกใช้บริการจากระบบ โดยโปรแกรมจะกลับมาที่ DEBUG เมื่อทำงานถึงบรรทัดถัดไป

การตั้งค่าให้กับรีจิสเตอร์ IP

เนื่องจากโปรแกรม DEBUG จะประมวลผลคำสั่งที่ตำแหน่ง CS:IP เสมอ แต่เมื่อโปรแกรมทำงานเสร็จรีจิสเตอร์ IP จะชี้ที่ออฟเซ็ตของคำสั่งถัดไปซึ่งเป็นคำสั่งที่ไม่ใช่ส่วนของโปรแกรมของเรา ดังนั้นภายหลังการทำงานของโปรแกรม เราควรจะตรวจสอบว่ารีจิสเตอร์ IP ชี้ไปยังจุดเริ่มต้นของโปรแกรมที่เราเขียนไว้หรือไม่ ซึ่งโดยปกติแล้วโปรแกรมจะเริ่มต้นที่ออฟเซ็ต 100h ถ้า รีจิสเตอร์ IP ชี้ไปที่อื่น เราสามารถตั้งค่าให้บรีจิสเตอร์ IP ได้โดยใช้คำสั่ง R

```
-RIP
IP 0106
:100
-
```

ตัวอย่างการทดลองโปรแกรมภาษาแอสเซมบลี

เราจะยกตัวอย่างการทดลองโปรแกรมภาษาแอสเซมบลีต่อไปนี้ด้วยโปรแกรม DEBUG

ตัวอย่างโปรแกรม

```
MOV    AX,5678h
MOV    BX,204h
MOV    CX,1234h
MOV    [200h],CX
MOV    [202h],AH
MOV    [203h],AL
MOV    [BX],AX
MOV    DX,[202h]
MOV    [204h],10h
```

ป้อนโปรแกรม

เราสามารถที่จะป้อนโปรแกรมและสั่งให้ DEBUG แปลโปรแกรมลงในหน่วยความจำโดยใช้คำสั่ง A ถ้าเราไม่ระบุแอดเดรสเริ่มต้นในครั้งแรกโปรแกรมจะถูกแปลลงในตำแหน่ง CS:100h และถ้าเป็นการสั่งครั้งถัดไปโปรแกรมจะแปลลงไปต่อเนื่องกับการสั่งครั้งก่อน

```
-A100
14FF:0100 mov ax,5678
14FF:0103 mov bx,204
14FF:0106 mov cx,1234
14FF:0109 mov [200],cx
14FF:010D mov [202],ah
14FF:0111 mov [203],al
14FF:0114 mov [bx],ax
14FF:0116 mov dx,[202]
14FF:011A mov [204],10
                  ^ Error
```

ในระหว่างการป้อนโปรแกรมอาจมีข้อผิดพลาดขึ้นโปรแกรม DEBUG จะแสดงข้อความขึ้นดังตัวอย่าง จากโปรแกรมดังกล่าวเราจะต้องระบุขนาดของการคัดลอกข้อมูลด้วย เราสามารถใส่คำสั่งที่แก้ไขแล้วลงไปใหม่ได้ทันที

```
14FF:011A mov word ptr [204],10
14FF:0120
-
```

เมื่อใส่โปรแกรมเสร็จจะต้องกดปุ่ม Enter ว่าง ๆ ไปเพื่อบอกโปรแกรม DEBUG ว่าโปรแกรมสิ้นสุดแล้ว เราสามารถเรียกดูโปรแกรมที่เราใส่ลงไปได้ด้วยคำสั่ง U

```
-U
14FF:0100 B87856                MOV     AX,5678
14FF:0103 BB0402                MOV     BX,0204
14FF:0106 B93412                MOV     CX,1234
14FF:0109 890E0002          MOV     [0200],CX
14FF:010D 88260202          MOV     [0202],AH
14FF:0111 A20302                MOV     [0203],AL
14FF:0114 8907                  MOV     [BX],AX
14FF:0116 8B160202          MOV     DX,[0202]
14FF:011A C70604021000      MOV     WORD PTR [0204],0010
-
```

เราจะเห็นทั้งโปรแกรมภาษาแอสเซมบลีและโปรแกรมภาษาเครื่องที่แปลแล้ว

ติดตามการทำงานของโปรแกรม

เราสามารถสั่งให้โปรแกรมทำงานโดยใช้คำสั่ง G (go) และตรวจสอบค่าในหน่วยความจำได้ แต่โปรแกรมที่จะทดลองนั้นต้องมีการสั่งให้โปรแกรมจบการทำงาน มิฉะนั้นโปรแกรมจะทำงานเลยไปถึงโปรแกรมอื่น ๆ ที่อยู่ในหน่วยความจำ ดังนั้นเราอาจจะใส่คำสั่ง INT 20h ปิดท้ายโปรแกรมไว้เพื่อสั่งให้โปรแกรมจบการทำงาน จากตัวอย่างแอดเดรสถัดไปของคำสั่งคือออฟเซ็ตที่ 0120h (สามารถหาได้โดยใช้คำสั่ง U แล้วสั่งเกออฟเซ็ตของตคำสั่งอื่น ๆ ถัดจากโปรแกรมที่เราป้อน) ดังนั้นเราจะป้อนคำสั่ง INT 20h ลงไปที่แอดเดรสนี้

```
-A120
14FF:0120 int 20
14FF:0122
-U100
14FF:0100 B87856                MOV     AX,5678
14FF:0103 BB0402                MOV     BX,0204
                ...
14FF:011A C70604021000      MOV     WORD PTR [0204],0010
-U11A
14FF:011A C70604021000      MOV     WORD PTR [0204],0010
14FF:0120 CD20                  INT     20
                ...
-
```

เมื่อเราใส่คำสั่งให้โปรแกรมจบการทำงานแล้ว เราสามารถที่ใช้คำสั่ง G (go) เพื่อสั่งให้โปรแกรมทำงานได้ และใช้คำสั่ง D (dump) เพื่อสังเกตค่าในหน่วยความจำ หรือ คำสั่ง R (register) เพื่อสังเกตค่าในรีจิสเตอร์ได้

```
-G

Program terminated normally

-D200
14FF:0200  34 12 56 78 10 00 DA EB-04 9D F8 EB 02 9D F9 89   4.Vx............
14FF:0210  36 8A DB 5E 5F 5A 59 C3-53 51 52 57 56 9C E8 6E   6..^_ZY.SQRWV..n
14FF:0220  00 83 3E 7A DB 40 7D 57-8A F7 8B 1E 7A DB FF 06   ..>z.@}W....z...
14FF:0230  7A DB B8 BA D8 E8 95 00-C7 47 07 00 00 F6 C6 01   z........G......
14FF:0240  74 03 89 6F 07 89 4F 05-88 77 02 8B 36 84 DB 89   t..o..O..w..6...
14FF:0250  37 03 36 5C D8 2B F7 89-77 03 8B 36 8A DB 89 77   7.6\.+..w..6...w
14FF:0260  09 8B F7 8B 3E 84 DB 03-F9 3B 3E 80 DB 7D 15 2B   ....>....;>..}.+
14FF:0270  F9 FC F3 A4 B0 00 AA 89-3E 84 DB 9D F8 EB 0A B8   ........>.......
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0100   NV UP EI PL NZ NA PO NC
14FF:0100 B87856        MOV     AX,5678
-
```

สังเกตว่าเมื่อเรากลับมายังโปรแกรม DEBUG หลังจากโปรแกรมที่เราป้อนทำงานเรียบร้อยแล้ว รีจิสเตอร์ต่าง ๆ จะถูกกำหนดค่าเริ่มต้องใหม่รวมทั้งรีจิสเตอร์ IP ด้วย ดังนั้นเราสามารถสังเกตผลของการทำงานได้จากค่าหน่วยความจำเท่านั้น เราสามารถสั่งให้โปรแกรมทำงานทีละบรรทัดโดยใช้คำสั่ง T หรือ P เราจะต้องระวังในเรื่องของการตั้งค่าเริ่มต้นของรีจิสเตอร์ IP ให้มาชี้ที่ตำแหน่งเริ่มต้นของโปรแกรมด้วย

```
-T

AX=5678  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0103   NV UP EI PL NZ NA PO NC
14FF:0103 BB0402        MOV     BX,0204
-T
AX=5678  BX=0204  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0106   NV UP EI PL NZ NA PO NC
```

```
14FF:0106 B93412        MOV     CX,1234
-T

AX=5678  BX=0204  CX=1234  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0109   NV UP EI PL NZ NA PO NC
14FF:0109 890E0002      MOV     [0200],CX                       DS:0200=1234
-T

AX=5678  BX=0204  CX=1234  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=010D   NV UP EI PL NZ NA PO NC
14FF:010D 88260202      MOV     [0202],AH                       DS:0202=56
-T

AX=5678  BX=0204  CX=1234  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0111   NV UP EI PL NZ NA PO NC
14FF:0111 A20302        MOV     [0203],AL                       DS:0203=78
-T
                ...
-T
AX=5678  BX=0204  CX=1234  DX=7856  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0120   NV UP EI PL NZ NA PO NC
14FF:0120 CD20          INT     20

-P
Program terminated normally
-
```

สังเกตว่าในคำสั่งสุดท้ายเราใช้คำสั่ง P เพราะคำสั่ง INT 20h เป็นการเรียกใช้บริการของระบบ และโปรแกรมจะกระโดดไปทำงานที่โปรแกรมย่อยของระบบซึ่งอาจจะยาวมาก ถ้าเราใช้คำสั่ง T เราจะได้เห็นการกระโดดไปทำงานนี้ และเราสามารถใช้คำสั่ง G เพื่อให้โปรแกรมทำงานจนจบต่อจากนั้นก็ได้

```
-T
AX=5678  BX=0204  CX=1234  DX=7856  SP=FFE8  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=14FF  IP=0120   NV UP DI PL NZ NA PO NC
14FF:0120 CD20          INT     20
-T
AX=5678  BX=0204  CX=1234  DX=7856  SP=FFE2  BP=0000  SI=0000  DI=0000
```

```
DS=14FF  ES=14FF  SS=14FF  CS=00C9  IP=0FA8   NV UP DI PL NZ NA PO NC
00C9:0FA8 90            NOP
-T
AX=5678  BX=0204  CX=1234  DX=7856  SP=FFE2  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=00C9  IP=0FA9   NV UP DI PL NZ NA PO NC
00C9:0FA9 90            NOP
-T
AX=5678  BX=0204  CX=1234  DX=7856  SP=FFE2  BP=0000  SI=0000  DI=0000
DS=14FF  ES=14FF  SS=14FF  CS=00C9  IP=0FAA   NV UP DI PL NZ NA PO NC
00C9:0FAA E8DB00        CALL    1088
-G
Program terminated normally
-
```

ข้อสังเกต เราควรจะพิจารณาค่าของรีจิสเตอร์ CS และ IP ก่อนที่จะเริ่มสั่งให้โปรแกรมทำงานใหม่เสมอ โดยปกติในโปรแกรม DEBUG ค่าของรีจิสเตอร์ CS จะมีค่าเท่ากันกับเซกเมนต์รีจิสเตอร์ตัวอื่น ๆ (DS ES และ SS) และเรานิยมป้อนโปรแกรมที่จะทดลองลงในตำแหน่ง CS:100h

สรุป

คำสั่งภาษาแอสเซมบลีเบื้องต้น โดยจะเป็นคำสั่งเกี่ยวกับการโอนย้ายข้อมูล ซึ่งมีอยู่ด้วยกัน 2 แบบ คือ
1. การโอนย้ายข้อมูลระหว่างรีจิสเตอร์กับรีจิสเตอร์ เป็นการโอนย้ายข้อมูลระหว่างรีจิสเตอร์สามารถทำได้ถ้าขนาดของรีจิสเตอร์ทั้งคู่เท่ากัน แต่ยังมีเซกเมนต์รีจิสเตอร์บางตัวซึ่งไม่ควรเข้าไปกำหนดค่าโดยตรง เช่น CS หรือ SS
2. การโอนย้ายข้อมูลกับหน่วยความจำ โดยปกติการโอนย้ายข้อมูลกับหน่วยความจำนั้น เราจะระบุเฉพาะออฟเซ็ตของหน่วยความจำที่เราต้องการจะโอนย้ายข้อมูลด้วย ออฟเซ็ตนั้นจะถูกนำมาประกอบกับเซกเมนต์รีจิสเตอร์ DS เพื่อเป็นตำแหน่งในหน่วยความจำที่แท้จริงที่จะอ่านเขียนข้อมูลด้วย

ดีบัก (Debug) คือ โปรแกรมที่พัฒนาเพื่อแก้ไขปัญหาพื้นฐานในระบบปฏิบัติการดอส (DOS = Disk Operation System) เป็นโปรแกรมสำหรับแก้ไขแฟ้มอย่างง่าย เป็นคำสั่งภายนอก (External Command) ของดอส (DOS) ที่นิยมใช้งานในกลุ่มนักพัฒนามาตั้งแต่ยุคระบบปฏิบัติการดอส เครื่องมือในการทดลองการโปรแกรมภาษาแอสเซมบลีก็คือโปรแกรม DEBUG เป็นโปรแกรมสารพัดประโยชน์สำหรับการทดลองเกี่ยวกับการเขียนโปรแกรมภาษาแอสเซมบลี โปรแกรมนี้มีอยู่ทั้งในระบบปฏิบัติการ DOS และ Windows

คำถามทบทวน

1. จงให้คำจำกัดความของคำว่าโอเปอร์แรนด์ (operand)
2. โปรแกรม DEBUG คืออะไรและมีหน้าที่อย่างไรในภาษาโปรแกรมภาษาแอสเซมบลี
3. รีจีสเตอร์ AX มีหน้าที่อย่างไรในคำสั่ง MOV
4. คำสั่งต่อไปนี้ทำงานอย่างไร
    - คำสั่ง G (go)
    - คำสั่ง D (dump)
    - คำสั่ง R (register)
5. คำสั่ง A (assemble) มีหน้าที่และทำงานอย่างไร
6. คำสั่งภาษาแอสเซมบลีที่ใช้สำหรับการโอนย้ายข้อมูลมีกี่แบบอะไรบ้าง