แผนการสอนประจำสัปดาห์ที่ 11

หัวข้อเนื้อหา การกระทำระดับบิต (The Bit Level)

รายละเอียด

การจัดการกับข้อมูลที่ได้ศึกษาในบทก่อน ๆ นั้นมีหน่วยย่อยในการจัดการเป็นไบต์ โดยจะไม่สามารถจัดการข้อมูลที่มีขนาดย่อยกว่านั้นได้ แต่ในการทำงานจริงในบางครั้งรูปแบบของข้อมูลที่นำไปใช้ จำเป็นต้องจัดการให้อยู่ในรูปของบิต ดังนั้นการใช้คำสั่งเกี่ยวกับการจัดการระดับบิตในการประมวลผลข้อมูลกลุ่ม ในบางกรณีคำสั่งกระทำระดับบิตสามารถช่วยให้การคำนวณต่าง ๆ ทำได้ง่ายและมีประสิทธิภาพมาขึ้นตามไปด้วย

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 11 การจัดการเรียนการสอน จะเกี่ยวข้องกับคำสั่งทางตรรกศาสตร์ การประยุกต์ใช้งานคำสั่งทางตรรกศาสตร์ การใช้คำสั่งเลื่อนบิต ตัวอย่างการใช้งานคำสั่งเลื่อนบิต การประยุกต์ใช้งานคำสั่งเลื่อนบิต คำสั่งหมุนบิต คำสั่งหมุนบิตที่ผ่านแฟล็กทด ตัวอย่างการใช้งานคำสั่งเกี่ยวกับการประมวลผลระดับบิต ซึ่งปัญหาของการจัดการกับข้อมูลจะกระทำในระดับไบต์ โดยจะไม่สามารถจัดการข้อมูลที่มีขนาดย่อยกว่านั้นได้ แต่ในการทำงานจริงในบางครั้งรูปแบบของข้อมูลที่นำไปใช้ จำเป็นต้องจัดการให้อยู่ในรูปของบิต ดังนั้นการใช้คำสั่งเกี่ยวกับการจัดการระดับบิตในการประมวลผลข้อมูลกลุ่ม ในบางกรณีคำสั่งกระทำระดับบิตสามารถช่วยให้การคำนวณต่าง ๆ ทำได้ง่ายและมีประสิทธิภาพมาขึ้นตามไปด้วย

11.1 คำสั่งทางตรรกศาสตร์

คำสั่งในกลุ่มนี้เป็นคำสั่งประมวลผลข้อมูลระดับบิต โดยจะนำค่าในแต่ละบิตของข้อมูลมาประมวลผลทางตรรกศาสตร์ คำสั่งในกลุ่มนี้ได้แก่ คำสั่ง AND คำสั่ง OR คำสั่ง XOR และคำสั่ง NOT ซึ่งรูปแบบการใช้งานของกลุ่มคำสั่งดังกล่าวจะมีลักษณะเหมือนกัน คือจะมีการ

รับโอเปอร์แรนด์สองตัว และจะนำข้อมูลในโอเปอร์แรนด์ตัวแรกมากระทำกับข้อมูลตัวที่สอง และจะเก็บผลลัพธ์ของการกระทำนั้นในโอเปอร์แรนด์ตัวแรก ส่วนในกรณีของคำสั่ง NOT จะรับโอเปอร์แรนด์ตัวเดียวและจะทำการกลับค่าในบิตแล้วเก็บผลลัพธ์ลงในโอเปอร์แรนด์ตัวนั้นเลย ตารางค่าความจริงของการกระทำทางตรรกศาสตร์ แสดงได้ดังตารางที่ 11.1

ตารางที่ 11.1 แสดงค่าของการกระทำทางตรรกศาสตร์

A	B	A and B	A or B	A xor B	not B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

คำสั่ง AND

ผลลัพธ์ของคำสั่ง AND จะมีบิตที่เป็น 1 เมื่อบิตของข้อมูลตัวตั้งทั้งสองตัวมีค่าเป็น 1 แสดงได้ (ตารางที่ 11.1)

ตัวอย่างคำสั่ง

```
mov     ax,1234h
and     ax,2345h        ; ax = (0001 0010 0011 0100) and (0010 0011 0100 0101)
                        ; ax = (0000 0010 0000 0100) = 0204h
mov     al,25h
and     al,0Fh          ; al = 05h
mov     bl,0AAh
and     bl,80h          ; al = 80h
```

คำสั่ง OR

ผลลัพธ์ของคำสั่ง OR จะมีบิตที่เป็น 1 เมื่อบิตของข้อมูลตัวตั้งตัวใดตัวหนึ่งหรือทั้งสองตัวมีค่าเป็น 1 แสดงได้ (ตารางที่ 11.1)

ตัวอย่างคำสั่ง

```
mov     ax,1234h
or      ax,2345h        ; ax = (0001 0010 0011 0100 and (0010 0011 0100 0101)
                        ; ax = (0011 0011 0111 0101) = 3375h
```

```
mov     al,25h
or      al,0Fh          ; al = 2Fh
mov     bl,55h
or      bl,80h          ; bl = 0D5h
```

คำสั่ง XOR

การทำงานของคำสั่ง XOR จะคล้ายกับคำสั่ง OR แต่ในกรณีที่ข้อมูลมีบิตที่เป็นหนึ่งทั้งคู่ ผลลัพธ์ที่ได้จะมีค่าเป็นศูนย์ (ตารางที่ 11.1) ลักษณะของการ XOR จะคล้ายกับการพิจารณาเหตุการณ์ที่เป็นไปได้ทั้งสองเหตุการณ์ แต่ไม่สามารถเป็นจริงพร้อมกันได้

ตัวอย่างคำสั่ง

```
mov     ax,1234h
xor     ax,2345h        ; ax = (0001 0010 0011 0100 and (0010 0011 0100 0101)
                        ; ax = (0011 0001 0111 0001) = 3171h
mov     al,25h
xor     al,0Fh          ; al = 2Ah
mov     bl,15h
xor     bl,35h          ; bl = 20h
```

คำสั่ง NOT

คำสั่ง NOT จะสลับบิตของโอเปอร์แรนด์จาก 0 เป็น 1 และ 1 เป็น 0 แสดงได้ (ตารางที่ 11.1)

ตัวอย่างคำสั่ง

```
mov     ax,1234h
not     ax              ; ax = not(0001 0010 0011 0100)
                        ; ax = (1110 1101 1100 1011) = 0EDCBh
```

คำสั่ง TEST

คำสั่ง TEST จะทำงานเหมือนคำสั่ง AND ทุกประการ แต่ผลลัพธ์จากการ AND จะไม่เขียนค่าลงในโอเปอร์แรนด์ตัวแรก ผลจากการใช้คำสั่งนี้จะปรากฏในแฟล็ก นิยมใช้คำสั่งนี้ในการทดสอบว่าข้อมูลในบิตที่ต้องการมีค่าเป็นหนึ่งหรือไม่ โดยเราจะพิจารณาผลลัพธ์จากแฟล็กทด

ตัวอย่างคำสั่ง

```
mov     al,7Ah
test    al,80h          ; test for the 7th bit
jz      biton           ; jump if 7th bit of al = 1
mov     bl,12h
test    bl,1            ; test for the 0th bit
```

การประยุกต์ใช้งานคำสั่งทางตรรกศาสตร์

การนำคำสั่งทางตรรกศาสตร์มาใช้ในการประมวลผลข้อมูลระดับบิตได้ จากตารางที่ 11.1 สามารถนำมาสร้างตารางที่ 11.2 ซึ่งแสดงผลของการใช้คำสั่งทางตรรกศาสตร์กับข้อมูลได้

ตารางที่ 11.2 แสดงผลของการใช้คำสั่งทางตรรกศาสตร์กับข้อมูล

B	A and B	A or B	A xor B
0	0	A	A
1	A	1	~A

จากตารางเราจะพบว่าถ้าเราต้องการให้บิตใดของข้อมูลมีค่าเป็นหนึ่งโดยที่บิตอื่นมีค่าคงเดิม เราสามารถใช้คำสั่ง AND ได้ และถ้าเราต้องการจะทำให้บิตใดของข้อมูลมีค่าเป็นศูนย์โดยไม่มีผลกระทบกับแฟล็กอื่น ๆ เราสามารถใช้คำสั่ง OR สำหรับคำสั่ง XOR เราจะใช้ในกรณีที่ต้องการกลับบิตของข้อมูลจากศูนย์เป็นหนึ่ง

ตัวอย่างการประยุกต์ใช้งานคำสั่งทางตรรกศาสตร์

โปรแกรมตัวอย่างต่อไปนี้จะเปลี่ยนบิตที่ 1 และ 2 ของ AL ให้มีค่าเป็นศูนย์ (การนับบิตจะนับบิตที่มีนัยสำคัญต่ำสุดเป็นบิตที่ 0) และเปลี่ยนบิตที่ 4 และบิตที่ 6 ให้มีค่าเท่ากับ 1 พร้อมทั้งกลับบิตที่ 3 ให้มีค่าตรงกันข้าม การทำงานควรจะมี แสดงได้ดังภาพที่ 11.1

```
AL      XXXX XXXX
and     1111 1001
or      0101 0000
xor     0000 1000
result  X1X1 X00X
```

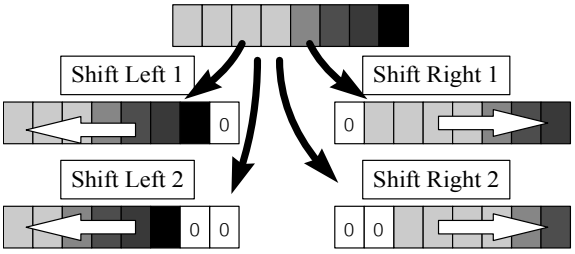
ภาพที่ 11.1 แสดงขั้นตอนการแปลงค่าของ AL

โปรแกรมจะมีลักษณะดังนี้

```
and     al,0F9h;clear bit 1&2
or      al,50h  ;set bit 4&6
xor     al,08h  ;switch bit 3
```

11.2 คำสั่งเลื่อนบิต

การประมวลผลอีกรูปแบบที่เราสามารถกระทำกับข้อมูลในระดับขั้นของบิตได้แก่ การเลื่อนบิต ลักษณะการเลื่อนบิต แสดงได้ดังภาพที่ 11.2 ในการเลื่อนบิตเราสามารถเลื่อนได้ทั้งทางซ้ายและทางขวา โดยคำสั่งสำหรับการเลื่อนบิตไปทางซ้ายได้แก่ **คำสั่ง SHL** (Shift Left) คำสั่งสำหรับการเลื่อนบิตไปทางขวาได้แก่ **คำสั่ง SHR** (Shift Right) เรานิยมใช้การเลื่อนบิตในการประมวลผลที่ต้องการประมวลผลข้อมูลทีละบิต และมีการประมวลผลเป็นแบบวงรอบ



ภาพที่ 11.2 แสดงลักษณะของการเลื่อนบิต

รูปแบบของคำสั่งเลื่อนบิตมีลักษณะดังนี้

SHR	*regs*,1	SHR	*mem*,1
SHR	*regs*,CL	SHR	*mem*,CL
SHR	*regs*,*number*	SHR	*mem*,*number*

โดยรูปแบบของคำสั่ง SHL จะมีลักษณะเหมือนคำสั่ง SHR รูปแบบที่สามจะใช้ได้กับหน่วยประมวลผล 80286 ขึ้นไปเท่านั้นโดยในการที่เราจะใช้รูปแบบของคำสั่งของ 80286 ในโปรแกรมเราจะต้องระบุ **คำสั่งเทียม 286** ลงในโปรแกรมด้วย โดยใส่คำสั่งนี้ก่อนหน้าการใช้งานคำสั่งครั้งแรก (ศัพท์บัญญัติ ราชบัณฑิตยสถาน, 2544)

ตัวอย่างการใช้งานคำสั่งเลื่อนบิต

โปรแกรมตัวอย่างต่อไปนี้เป็นโปรแกรมนับจำนวนบิตที่มีค่าเป็นหนึ่งใน AX โดยจะให้ผลลัพธ์ใน BL

```
        mov     bl,0
        mov     cx,16           ; 16 bits
procbits:
        test    ax,1            ; test for last bit
        jz      doprocbit       ; if last bit=0 jump
        inc     bl
doprocbit:
        shr     ax,1            ; next bit
        loop    procbits
```

ความหมายทางคณิตศาสตร์ของการเลื่อนบิต

ตารางที่ 11.3 แสดงผลลัพธ์ของการเลื่อนบิตของข้อมูลต่าง ๆ

จากตารางจะสังเกตได้ว่านอกจากการเลื่อนบิตจะมีความหมายโดยตรงคือการเลื่อนบิตไปทางซ้ายหรือทางขวาแล้ว การเลื่อนบิตยังมีความหมายทางคณิตศาสตร์อีกด้วย

ข้อมูล (ค่า)	เลื่อนบิตไปทาง	จำนวน (บิต)	ผลลัพธ์ (ค่า)
0010 1110 (46)	ซ้าย	1	0101 1100 (92)
0010 1110 (46)	ซ้าย	2	1011 1000 (184)
0010 1110 (46)	ซ้าย	3	0111 0000 (112)
0110 0100 (100)	ขวา	1	0011 0010 (50)
0110 0100 (100)	ขวา	2	0001 1001 (25)
0110 0100 (100)	ขวา	3	0000 1100 (12)

สังเกตว่าการเลื่อนบิตไปทางซ้ายจะมีผลลัพธ์เหมือนกับการคูณด้วยกำลังของสอง ยกตัวอย่างเช่น การเลื่อนบิตไปทางซ้าย 1 บิตจะเหมือนกับการคูณด้วยสอง ข้อสังเกตคือจะต้องพิจารณากรณีที่ข้อมูลอยู่ในขอบเขตด้วย เช่น กรณีของการเลื่อน 0010 1110 ไปทางซ้าย 3 บิต (คูณด้วย 8) ผลลัพธ์ที่ได้จะมีความผิดพลาด การเลื่อนบิตไปทางขวาจะให้ผลลัพธ์ตรงกันข้ามกับการเลื่อนบิตไปทางขวา นั่นคือจะเสมือนการหารด้วยกำลังสอง (สังเกตว่าผลลัพธ์ที่ได้จะมีการปัดเศษเนื่องจากบิตที่เลื่อนจะหายไป เช่นในตัวอย่างที่เลื่อนบิตทางขวา 3 บิต)

คำสั่งเลื่อนบิตแบบคิดเครื่องหมาย : คำสั่ง SAL และคำสั่ง SAR

ถ้าใช้การเลื่อนบิตแทนการคูณหรือหารด้วยกำลังของสองกับตัวเลขแบบคิดเครื่องหมาย เราจะพบว่าการเลื่อนบิตไปทางซ้ายที่แสดงถึงการคูณนั้นยังสามารถใช้กับตัวเลขแบบคิดเครื่องหมายได้ เนื่องจากหลักที่เลื่อนเข้ามาแทนนั้นยังคงเป็นเลขศูนย์เหมือนในกรณีของเลขไม่คิดเครื่องหมาย แต่ในกรณีของการเลื่อนบิตไปทางขวาที่ใช้สำหรับการหารด้วยกำลังของสองนั้น บิตที่เลื่อนเข้ามาแทนอาจมีค่าเป็น 0 หรือ 1 ก็ได้ขึ้นกับเครื่องหมายของตัวเลขนั้น จึงมีคำสั่งเลื่อนบิตที่ใช้สำหรับเลขที่มองเป็นเลขคิดเครื่องหมาย คือ **คำสั่ง SAL** (Shift Arithmetic Left) และ **คำสั่ง SAR** (Shift Arithmetic Right) คำสั่ง SAL จะทำงานเหมือนคำสั่ง SHL ทุกประการ ตัวอย่างการใช้งานคำสั่งเป็นดังตารางที่ 11.4

ตารางที่ 11.4 แสดงตัวอย่างผลลัพธ์ของการเลื่อนบิตแบบคิดเครื่องหมาย

ข้อมูล (ค่า)	เลื่อนบิตไปทาง	จำนวน (บิต)	ผลลัพธ์ (ค่า)
0010 1110 (46)	ซ้าย	1	0101 1100 (92)
1110 1000 (–24)	ซ้าย	2	1101 0000 (–48)
0110 0100 (100)	ขวา	1	0011 0010 (50)
1010 1100 (–84)	ขวา	2	1110 1011 (–21)
1010 1100 (–84)	ขวา	3	1111 0101 (–11)

การประยุกต์ใช้งานคำสั่งเลื่อนบิต

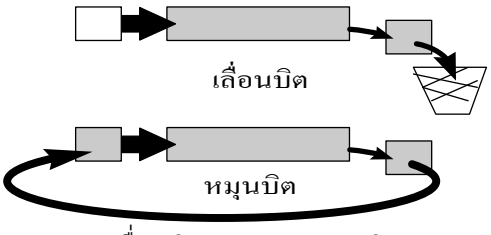
การนำคำสั่งเลื่อนบิตไปใช้ในการคูณและหารข้อมูลได้ โดยการใช้คำสั่งเลื่อนบิตแทนการใช้คำสั่ง MUL ทำให้การคูณทำงานได้เร็วขึ้นและในบางกรณีทำให้เขียนโปรแกรมได้ง่ายขึ้นด้วย

ตัวอย่างคำสั่ง

```
mov     bl,al
shl     al,1
add     bl,al           ;bl = al*3
mov     cl,2
shl     ax,cl
mov     bx,ax
shl     ax,1
add     bx,ax           ;bx=(ax*4)+(ax*8) = ax*12
```

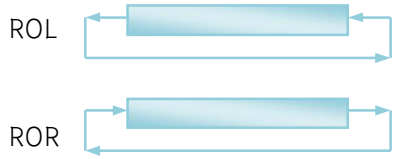
11.3 คำสั่งหมุนบิต

คำสั่งหมุนบิตมีความแตกต่างกับคำสั่งเลื่อนบิตในจุดที่ว่า บิตที่เลื่อนไปแล้วไม่ได้ถูกทิ้งหายไป แต่จะถูกนำมาใส่แทนบิตที่เลื่อนไป แสดงได้ดังภาพที่ 11.3



ภาพที่ 11.3 แสดงลักษณะของการเลื่อนบิต และ การหมุนบิต

เช่นเดียวกับคำสั่งเลื่อนบิต คำสั่งหมุนบิตมีลักษณะการหมุนสองแบบคือ หมุนไปทางซ้าย (**คำสั่ง ROL** : Rotate Left) และ หมุนไปทางขวา (**คำสั่ง ROR** : Rotate Right) รูปแบบของคำสั่งทั้งสองจะมีลักษณะเหมือนคำสั่งเลื่อนบิต การทำงานของคำสั่งทั้งสอง แสดงได้ดังภาพที่ 11.4



ภาพที่ 11.4 แสดงลักษณะการทำงานของคำสั่งหมุนบิต

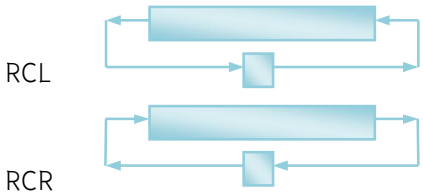
นิยมใช้คำสั่งหมุนบิตแทนคำสั่งเลื่อนบิตในกรณีที่ต้องการให้ค่าของข้อมูลกลับเหมือนเดิมหลังประมวลผลครบรอบ

ตัวอย่างคำสั่ง

```
        mov     al,4Ah
        mov     cl,4
        ror     al,cl           ;al = 0A4h
        mov     bx,92EAh        ;bx=1001 0010 1110 1010
        rol     bx,1            ;bx=0010 0101 1101 0101=35D5h
        mov     cx,8
        mov     dx,0
        loophere:
                xor     dx,ax
                rol     ax,1
                loop    loophere
```

คำสั่งหมุนบิตที่ผ่านแฟล็กทด

คำสั่งหมุนบิตอีกกลุ่มหนึ่งจะเป็นการหมุนโดยนำบิตไปผ่านแฟล็กทด ลักษณะการทำงาน แสดงได้ดังภาพที่ 11.5 ซึ่งจะเห็นว่าบิตที่เข้ามาแทนบิตที่หมุนไปจะนำมาจากแฟล็กทด และบิตที่ถูกหมุนออกไปจะเข้าไปแทนค่าในแฟล็กทด โดยคำสั่งหมุนบิตผ่านแฟล็กทดคือ**คำสั่ง RCL** (Rotate Carry Left) และ**คำสั่ง RCR** (Rotate Carry Right)



ภาพที่ 11.5 แสดงลักษณะการทำงานของคำสั่งหมุนบิตที่ผ่านแฟล็กทด

สังเกตว่าบิตที่ล้นออกมาจะถูกนำไปพักที่แฟล็กทด ก่อนที่จะนำมาแทนที่ในข้อมูล นิยมใช้คำสั่งหมุนบิตผ่านแฟล็กทดในการเลื่อนบิตข้อมูลที่เก็บต่อเนื่องอยู่ในหลายรีจิสเตอร์ ในการใช้งานคำสั่งนี้เราจะต้องกำหนดค่าให้กับแฟล็กทดเสียก่อน โดยใช้คำสั่ง STC และคำสั่ง CLC

ตัวอย่างการใช้งานคำสั่งหมุนบิตที่ผ่านแฟล็กทด

แสดงการเขียนคำสั่งแสดงการเลื่อนบิตของข้อมูลขนาด 32 บิตที่อยู่ในรีจิสเตอร์ DX,AX ไปทางซ้าย 1 บิต

```
clc
rcl     ax,1
rcl     dx,1
```

ตัวอย่างคำสั่ง การคูณข้อมูลขนาด 48 บิตที่เก็บในรีจิสเตอร์ BX,DX และ AX ต่อเนื่องกัน ด้วย 4

```
clc
rcl     ax,1
rcl     dx,1
rcl     bx,1
clc
rcl     ax,1
rcl     dx,1
rcl     bx,1
```

ตัวอย่างคำสั่ง การหารข้อมูลใน DX,AX ด้วย 2

```
clc
rcr     dx,1
rcr     ax,1
```

ตัวอย่างการใช้งานคำสั่งเกี่ยวกับการประมวลผลระดับบิต

โปรแกรมตัวอย่างต่อไปนี้เป็นโปรแกรมที่แสดงค่ารหัสแอสกีของปุ่มที่รับจากผู้ใช้เป็นเลขฐานสอง โปรแกรมย่อยที่แสดงข้อมูลเป็นรหัสเลขฐานสองใช้การเลื่อนบิตในการประมวลผล

```
        .model   small
        .dosseg
        .code
        ; Display Binary (input : al)
        dispbin  proc  near
                push  ax
                push  cx
                push  dx
                mov   cx,8
        printloop:
                test  al,80h            ;test for 1
                jz    printzero
                mov   dl,'1'
                jmp   printit
        printzero:
                mov   dl,'0'
        printit: mov   ah,2
                int   21h
                shl   al,1
                loop  printloop
                pop   dx
                pop   cx
                pop   ax
                ret
        dispbin  endp
        start:
                mov   ah,1
                int   21h
                call  dispbin
                mov   ax,4C00h
                int   21h
        end      start
```

สรุป

คำสั่งทางตรรกศาสตร์เป็นคำสั่งประมวลผลข้อมูลระดับบิต โดยจะนำค่าในแต่ละบิตของข้อมูลมาประมวลผลทางตรรกศาสตร์ คำสั่งในกลุ่มนี้ได้แก่ คำสั่ง AND คำสั่ง OR คำสั่ง XOR และคำสั่ง NOT รูปแบบการใช้งานของคำสั่ง AND คำสั่ง OR และคำสั่ง XOR จะมีลักษณะเหมือนกัน คือจะรับโอเปอร์แรนด์สองตัว และจะนำข้อมูลในโอเปอร์แรนด์ตัวแรกมากระทำกับข้อมูลตัวที่สอง และจะเก็บผลลัพธ์ของการกระทำนั้นในโอเปอร์แรนด์ตัวแรก ส่วนในกรณีของคำสั่ง NOT จะรับโอเปอร์แรนด์ตัวเดียว และจะทำการกลับค่าในบิตแล้วเก็บผลลัพธ์ลงในโอเปอร์แรนด์ตัวนั้นเลย ส่วนการประมวลผลอีกรูปแบบที่เราสามารถกระทำกับข้อมูลในระดับขั้นของบิตได้แก่การเลื่อนบิต ในการเลื่อนบิตเราสามารถเลื่อนได้ทั้งทางซ้ายและทางขวา โดยคำสั่งสำหรับการเลื่อนบิตไปทางซ้ายได้แก่ คำสั่ง SHL (Shift Left) คำสั่งสำหรับการเลื่อนบิตไปทางขวาได้แก่ คำสั่ง SHR (Shift Right) เรานิยมใช้การเลื่อนบิตในการประมวลผลที่ต้องการประมวลผลข้อมูลทีละบิต และมีการประมวลผลเป็นแบบวงรอบ

คำถามทบทวน

1. จงแสดงผลลัพธ์ของคำสั่งจากคำสั่งข้างล่างที่ให้มาเมื่อบิตของข้อมูลตัวตั้งทั้งสองตัวมีค่าเป็น 1

```
1.1     mov     ax,2345h
        and     ax,3456h        ; ax = ? and ?
                                ; ax = ?
1.2     mov     ax,2345h
        or      ax,3456h        ; ax = ? and ?
                                ; ax = ?
1.3     mov     ax,2345h
        xor     ax,3456h        ; ax = ? and ?
                                ; ax = ?
```

2.จงแสดงผลลัพธ์ของการเลื่อนบิตของข้อมูลต่าง ๆ จากตารางที่ให้มา

ข้อมูล (ค่า)	เลื่อนบิตไปทาง	จำนวน(บิต)	ผลลัพธ์ (ค่า)
0011 1110 (63)	ซ้าย	1	?
0011 1110 (63)	ซ้าย	2	?
0011 1110 (63)	ซ้าย	3	?
0111 0100 (116)	ขวา	1	?
0111 0100 (116)	ขวา	2	?
0111 0100 (116)	ขวา	3	?

3.จงแสดงผลลัพธ์ของการเลื่อนบิตแบบคิดเครื่องหมายของข้อมูลต่าง ๆ จากตารางที่ให้มา

ข้อมูล (ค่า)	เลื่อนบิตไปทาง	จำนวน(บิต)	ผลลัพธ์ (ค่า)
1100 0010 (–63)	ซ้าย	1	?
1101 1000 (–40)	ซ้าย	2	?
1100 0010 (–63)	ขวา	1	?
1000 1100 (–116)	ขวา	2	?
1000 1100 (–116)	ขวา	3	?

4.จงอธิบายพร้อมยกตัวอย่างคำสั่งหมุนบิตผ่านแฟล็กทดต่อไปนี้

4.1 คำสั่ง RCL (Rotate Carry Left)

4.2 คำสั่ง RCR (Rotate Carry Right)

เอกสารอ้างอิง

ราชบัณฑิตยสถาน. (2544). *ศัพท์บัญญัติ ราชบัณฑิตยสถาน.* ค้นเมื่อ 15 มิถุนายน 2557,จาก :http://rirs3.royin.go.th/coinages/

การเลื่อนบิตของข้อมูล (2557). *วิกิพีเดีย สารานุกรมเสรี.* ค้นเมื่อ มิถุนายน 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.

.