ภาษา Assembly

AND

OR

XOR

ภาษา Assembly

คือ ภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งจะทำงานโดยขึ้นกับรุ่นของไมโครโพรเซสเซอร์ หรือหน่วยประมวลผล (CPU) ของเครื่องคอมพิวเตอร์ การใช้ภาษาแอสเซมบลีจำเป็นต้องผ่านการแปลภาษาด้วยคอมไพเลอร์เฉพาะเรียกว่า แอสเซมเบลอร์ (assembler) ให้อยู่ในรูปของรหัสคำสั่งก่อน (เช่น .OBJ) โดยปกติ ภาษานี้ค่อนข้างมีความยุ่งยากในการใช้งาน และการเขียนโปรแกรมเป็นจำนวนบรรทัดมากมากกว่า เมื่อเปรียบเทียบกับการใช้ภาษาระดับสูง เช่น ภาษา C หรือภาษา BASIC แต่จะทำให้ได้ผลลัพธ์การทำงานของโปรแกรมเร็วกว่า และขนาดของตัวโปรแกรมมีขนาดเนื้อที่น้อยกว่าโปรแกรมที่สร้างจากภาษาอื่นมาก จึงนิยมใช้ภาษานี้ เมื่อต้องการประหยัดเวลาทำงานของเครื่องคอมพิวเตอร์ และเพิ่มประสิทธิภาพของโปรแกรม

โครงสร้างของโปรแกรมภาษาแอสแซมบลี

ประกอบด้วย 4 ส่วนหลักคือ

01 ลาเบล (Label)

ใช้ในการอ้างถึงบรรทัดใดบรรทัดหนึ่งของโปรแกรมที่ทำการเขียนขึ้น

02 รหัสนีโมนิก (Mnemonic)

เป็นส่วนแสดงคำสั่งของไมโครคอนโทรลเลอร์ที่ต้องการให้กระทำ

03 โอเปอร์แรนด์ (Operand)

เป็นส่วนที่แสดงถึงตัวกระทำหรือถูกกระทำและข้อมูลที่ใช้ในการกระทำตามคำสั่งที่กำหนดโดยรหัสนีโมนิกก่อนหน้านี้

04 คอมเมนต์ (Comment)

เป็นส่วนที่ผู้เขียนโปรแกรมเขียนขึ้นเพื่อใช้ในการอธิบายคำสั่งที่กระทำหรือผลของการกระทำคำสั่งในบรรทัดหรือโปรแกรมย่อยนั้นๆ ทั้งนี้เพื่อช่วยให้ผู้เขียนสามารถตรวจสอบโปรแกรมที่เขียนขึ้นได้ง่ายรวมถึงเป็นประโยชน์ต่อผู้ที่นำโปรแกรมนั้นมาศึกษาใหม่อีกด้วย

คำสั่งทางตรรกศาสตร

คำสั่งในกลุ่มนี้เป็นคำสั่งประมวลผลข้อมูลระดับบิต โดยจะนำค่าในแต่ละบิตของ ข้อมูลมาประมวลผลทางตรรกศาสตร์ คำสั่งในกลุ่มนี้ได้แก่ คำสั่ง AND คำสั่ง OR คำสั่ง XOR และคำสั่ง NOT ซึ่งรูปแบบการใช้งานของกลุ่มคำสั่งดังกล่าวจะมีลักษณะเหมือนกัน คือ จะมีการ รับโอเปอร์แรนด์สองตัว และจะนำข้อมูลในโอเปอร์แรนด์ตัวแรกมากระทำกับข้อมูลตัวที่สอง และจะเก็บผลลัพธ์ของการกระทำนั้นในโอเปอร์แรนด์ตัวแรก ส่วนในกรณีของคำสั่ง NOT จะ รับโอเปอร์แรนด์ตัวเดียวและจะทำการกลับค่าในบิตแล้วเก็บผลลัพธ์ลงในโอเปอร์แรนด์ตัวนั้น เลย ตารางค่าความจริงของการกระทำทางตรรกศาสตร

คำสั่ง AND

ผลลัพธ์ของคำสั่ง AND จะมีบิตที่เป็น 1 เมื่อบิตของข้อมูลตัวตั้งทั้งสองตัวมีค่าเป็น 1

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	

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

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	

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 ลักษณะของการ XOR จะคล้ายกับการพิจารณา เหตุการณ์ที่เป็นไปได้ทั้งสองเหตุการณ์ แต่ไม่สามารถเป็นจริงพร้อมกันได้

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	

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
```

สรุป

คำสั่งทางตรรกศาสตร์เป็นคำสั่งประมวลผลข้อมูลระดับบิต โดยจะนำค่าในแต่ละ บิตของข้อมูลมาประมวลผลทางตรรกศาสตร์ คำสั่งในกลุ่มนี้ได้แก่ คำสั่ง AND คำสั่ง OR คำสั่ง XOR และคำสั่ง NOT รูปแบบการใช้งานของคำสั่ง AND คำสั่ง OR และคำสั่ง XOR จะมี ลักษณะเหมือนกัน คือจะรับโอเปอร์แรนด์สองตัว และจะนำข้อมูลในโอเปอร์แรนด์ตัวแรกมา กระทำกับข้อมูลตัวที่สอง และจะเก็บผลลัพธ์ของการกระทำนั้นในโอเปอร์แรนด์ตัวแรก ส่วนใน กรณีของคำสั่ง NOT จะรับโอเปอร์แรนด์ตัวเดียว และจะทำการกลับค่าในบิตแล้วเก็บผลลัพธ์ ลงในโอเปอร์แรนด์ตัวนั้นเลย ส่วนการประมวลผลอีกรูปแบบที่เราสามารถกระทำกับข้อมูลใน ระดับชั้นของบิต ได้แก่การเลื่อนบิต ในการเลื่อนบิตเราสามารถเลื่อนได้ทั้งทางซ้ายและทางขวา โดยคำสั่งสำหรับการเลื่อนบิตไปทางซ้ายได้แก่ คำสั่ง SHL (Shift Left) คำสั่งสำหรับการเลื่อน บิตไปทางขวาได้แก่ คำสั่ง SHR (Shift Right) เรานิยมใช้การเลื่อนบิตในการประมวลผลที่ ต้องการประมวลผลข้อมูลทีละบิต และมีการประมวลผลเป็นแบบวงรอบ

ที่มา : http://comsci.sci.dusit.ac.th/wp-content/uploads/2016/10/Book_AM257.pdf