บทที่ 2

ไมโครโปรเซสเซอร์-แอสเซมเบลอร์

2.1 เปรียบเทียบระดับภาษา (Comparision of Language Levels)[1]

2.1.1 การทำโปรแกรมภาษาเครื่อง (Machine Language Programming)

คำสั่งที่คอมพิวเตอร์สามารถทำงานได้จะต้องอยู่ในรูปของเลขฐานสอง ซึ่งซีพียู (CPU) จะทำการดึงมาจากหน่วยความจำ แล้วทำการแปลและปฏิบัติตามคำสั่งนั้น ตัวอย่างของเลขฐานสอง 8 บิต เช่น

10000000

เราอาจเข้าใจได้หลายอย่าง เช่น เป็นค่า คอมพลีเมนท์ของสอง ของ -128 หรือค่า 128 เป็นต้น คอมพิวเตอร์จะทราบว่าอะไรคือตำแหน่งในส่วนความจำ หรือส่วนไหนเป็นคำสั่ง ก็แล้วแต่การใช้คำสั่งนั้นๆ การเขียนโปรแกรมในลักษณะนี้ คือเขียนโปรแกรมในรูปของเลขฐานสอง เรียกว่า การทำโปรแกรมภาษาเครื่อง ซึ่งการเขียนโปรแกรมแบบนี้ทำให้เกิดความลำบาก เพราะข้อมูลหรือคำสั่งก็มีลักษณะเหมือนกันดังตัวอย่างในรูปที่ 2.1

Address	Contents
0	00100001
1	01000000
2	00000000
3	00010001
4	01100000
5	00000000
6	00000110
7	00001010
8	01111110
9	00010010
10	00100011
11	00010011
12	00000101
13	11000010
14	00001000
15	00000000
16	01110110

รูปที่ 2.1 โปรแกรมภาษาเครื่อง

---

1 Lance A. Leventhal in Introduction to Microproccsors pp. 127-165, Prentice/Hall International, 1978.

เพื่อให้ผู้เขียนโปรแกรมเขียนโปรแกรมได้ละดวกยิ้น และสามารถทำความเข้าใจโปรแกรมได้ง่ายยิ้น จึงได้มีการตั้งชื่อแทน รีจิสเตอร์, ข้อมูล หรือ ตำแหน่งแหล่งที่ของหน่วย - ความจำ ในการเขียนโปรแกรม แล้วจึงค่อยทำการเปลี่ยนชื่อต่างๆนั้นให้อยู่ในรูปของเลขฐานสอง โดยใช้ตารางคำสั่งของบริษัทที่ขายไมโครโปรเซสเซอร์เบอร์นั้นๆ ชื่อต่างๆที่ใช้เช่น

ADD - add
SUB - Subtract
LD - Load
JMP - Jump
A - Accumulator
X - Index register

วิธีการที่เขียนโปรแกรมแบบนี้ เรียกว่า การทำโปรแกรมภาษาแอสเซมบลี (Assembly Language Programming)และวิธีการที่ทำการเปลี่ยนโปรแกรมไปอยู่ในรูปของเลขฐาน-สอง เรียกว่า การทำแอสเซมบลีด้วยมือ (Hand Assembly)

```
        LDAB   $41      COUNT =LENGTH OF ARRAY
        DECB
        LDAA   $42
        LDX    #$43
MAXM    CMPA   X        IS MAXIMUM LARGER THAN CURRENT ENTRY?
        BCC    NOCHG    YES, KEEP MAXIMUM
        LDAA   X        NO, REPLACE WITH CURRENT ENTRY
NOCHG   INX
        DECB
        BNE    MAXM
        STA    $40      SAVE MAXIMUM
        SWI
```

รูปที่ 2.2 ตัวอย่างโปรแกรมแอสเซมบลี

2.1.2 มายีนแอล่เซมบลีและแอล่เซมเบลอร์โปรแกรม

การทำแอล่เซมบลีด้วยมือนั้นถึงแม้ว่าจะไม่ใช่เรื่องยาก เช่นการเปลี่ยน รหัล์คำสั่ง หรือเลขฐานสิบ ไปเป็นเลขฐานส่อง หรือการคำนวณหาตำแหน่งของหน่วยความจำ แต่จะต้องใช้เวลาในการทำงานเหล่านี้มาก ทั้งยังอาจจะเกิดความผิดพลาดได้ง่ายด้วย เพราะต้องมีการลงรหัล์ตัวเลขเป็นจำนวนมาก จึงควรให้คอมพิวเตอร์ช่วยแก้ปัญหานี้โดยการใช้โปรแกรม

วิธีการลดความผิดพลาดอย่างหนึ่งทำได้โดยการลดจำนวนการลงรหัล์ตัวเลข คือแทนที่จะเขียนโปรแกรมให้อยู่ในรูปของเลขฐานส่องเลย เราก็เปลี่ยนเขียนให้อยู่ในรูปของเลขฐานแปด หรือเลขฐานสิบหก จะเห็นว่าจะทำให้ลดการเขียนตัวเลขไปถึง 2 ใน 3 หรือ 1 ใน 3 แต่เราจะต้องทำให้คอมพิวเตอร์ล่ามารถรับเลขฐานแปดหรือฐานสิบหก โดยการเขียนโปรแกรมเพื่อใช้ในการเปลี่ยนเลขเหล่านี้ให้อยู่ในรูปของเลขฐานส่อง ซึ่งส่วนนี้ส่วนใหญ่จะเป็นโปรแกรมที่อยู่ในรอม (ROM)

คอมพิวเตอร์ล่ามารถช่วยเราได้มากกว่าการเปลี่ยนเลขฐานแปด หรือ เลขฐานสิบหก ไปเป็นเลขฐานส่องเท่านั้น โดยเราล่ามารถใช้คอมพิวเตอร์ทำงานแทนการทำแอล่เซมบลีด้วยมือ ด้วยการใช้โปรแกรมที่เรียกว่า แอล่เซมเบลอร์ แอล่เซมเบลอร์นี้จะทำหน้าที่ในการเปลี่ยนรหัล์นีโมนิค ไปให้อยู่ในรูปของเลขฐานส่อง และล่ร้างตารางสัญญลักษณ์ ซึ่งจะล่ามารถทำงานได้รวดเร็วและถูกต้องกว่าการทำแอล่เซมบลีด้วยมือมาก แอล่เซมเบลอร์นี้จะทำเล่มือนว่าโปรแกรมที่เราเขียนขึ้นเป็นข้อมูลเข้าแล้วจะทำการแปลเป็นรหัล์เครื่องออกมาให้,โปรแกรมแอล่เซมเบลอร์นี้เพื่อไม่ให้ขึ้นอยู่กับโครงล่ร้างของเครื่องคอมพิวเตอร์มากเกินไป จึงควรเขียนอยู่ในรูปของภาษาระดับสูง เช่น ฟอร์แทรน, เบลิก เป็นต้น

2.2 โครงล่ร้างของแอล่เซมเบลอร์

วัตถุประล่งค์ของแอล่เซมเบลอร์ ก็เพื่อทำการแปลภาษาแอล่เซมบลี ให้ไปอยู่ในรูปของรหัล์เครื่อง ซึ่งแอล่เซมเบลอร์แต่ละตัวก็จะมีการทำงานมากน้อยต่างกันแล้วแต่ผู้ล่ร้าง แต่ก็ให้ประโยชน์มากกว่าการทำแอล่เซมบลีด้วยมืออยู่ดี โดยทั่วไปแล้วแอล่เซมเบลอร์ในปัจจุบันจะมีโครงล่ร้างต่างๆกัน เช่นการใช้เลเบล (Label) และการอ้างถึงตำแหน่งของหน่วยความจำโดยใช้สัญญลักษณ์ แอล่เซมเบลอร์ทั่วไปจะมีล่วนประกอบดังนี้

2.2.1 รูปแบบของคำสั่งแอล่เซมเบลอร์ (Assembler Instruction Format)

ในหนึ่งคำสั่งของแอล่เซมเบลอร์จะประกอบด้วยเขตข้อมูลต่างๆดังรูปที่ 2.3

เลเบล LABEL	รหัล์ดำเนินการ OPERATION CODE	ตำแหน่งที่อยู่ของข้อมูล ADDRESS	หมายเหตุ COMMENT

รูปที่ 2.3 รูปแบบของคำสั่งแอล่เซมเบลอร์

ตัวอย่างเช่น

LAST	JUMP	START	: RETURN TO BEGINNING OF PROGRAM

แอสเซมเบลอร์บางชนิดอาจจะใช้วิธีการกำหนดสดมภ์ เพื่อเป็นการแบ่งเขตข้อมูลโดยกำหนดว่าเขตข้อมูลใดจะเริ่มต้นที่สดมภ์ใด การใช้รูปแบบชนิดนี้เรียกว่า รูปแบบคงที่(Fixed format) การกำหนดเช่นนี้ทำให้การเขียนแอสเซมเบลอร์ง่ายขึ้นด้วย แต่โดยทั่วไปแล้วเพื่ออำนวยความสะดวกให้แก่ผู้เขียนโปรแกรม ไม่ต้องระมัดระวังในการใช้เขตข้อมูลแต่ละเขตให้ถูกต้องตามสดมภ์ที่กำหนดไว้ จึงได้มีการเขียนแอสเซมเบลอร์อีกแบบหนึ่งที่เรียกว่า ฟรี ฟอแม็ท(Free format) คือไม่มีการกำหนดว่าสดมภ์ใดเป็นของเขตข้อมูลใด การแบ่งของแต่ละเขตทำโดยการใช้เครื่องหมายเฉพาะเป็นตัวแบ่งเขต ซึ่งเรียกว่า ดิลิมิเตอร์(Delimiter) เช่นเครื่องหมายจุลภาค (,) เป็นต้น

2.2.2 เลเบล

การใช้เลเบลนี้ทำให้ผู้เขียนโปรแกรมสามารถอ้างถึงตำแหน่งต่างๆโดยการใช้ชื่อแทน แอสเซมเบลอร์ส่วนใหญ่จะยอมให้มีการใช้เลเบลนี้ โดยการกำหนด เป็นตัวเลข, ตัวอักษร หรือทั้งตัวอักษรและตัวเลข ซึ่งโดยทั่วไปจะประกอบด้วยตัวอักษรปนตัวเลขเป็นจำนวนไม่เกิน 6 ตัว ส่วนของเลเบลจะใช้ก็ต่อเมื่อมีประโยชน์ในการอ้างถึงตำแหน่งต่างๆในการเขียนโปรแกรมเท่านั้น ซึ่งโดยปกติจะปล่อยว่างไว้

2.2.3 รหัสดำเนินการแอสเซมเบลอร์ (Assembler Operation Codes)

ส่วนนี้จะเก็บอยู่ในรูปของรหัสนีโมนิคในเขตของรหัสดำเนินการ ซึ่งเขตข้อมูลนี้จะปล่อยว่างไว้ไม่ได้ในแต่ละคำสั่ง แอสเซมเบลอร์จะทำการเปรียบเทียบรหัสนีโมนิค กับตารางที่มีอยู่แล้วและจะให้ค่าเลขฐานสองที่มีค่าสอดคล้องกับรหัสนั้นๆ

2.2.4 รหัสดำเนินการเทียม (Pseudo Operation Codes)

รหัสดำเนินการเทียมนี้จะเขียนอยู่ในเขตของรหัสดำเนินการ รหัสดำเนินการเทียมนี้จะไม่มีการแปลไปเป็นเลขฐานสอง จะใช้เกี่ยวกับการกำหนดสัญญลักษณ์ การกำหนดเนื้อที่หน่วยความจำ เป็นต้น เช่น

ORIGIN	-	กำหนดจุดเริ่มต้นของโปรแกรม
EQUATE	-	กำหนดชื่อให้กับค่าคงที่
RESERVE	-	กำหนดพื้นที่ของหน่วยความจำ
DATA	-	การกำหนดข้อมูล
END	-	บอกการสิ้นสุดของโปรแกรม

2.2.5 ตารางสัญญลักษณ์ (Symbol Table)

เป็นตารางที่ใช้เก็บชื่อหรือเลเบลที่ใช้ในโปรแกรมและค่าของมัน เพื่อใช้ในการอ้างถึงในโปรแกรม

2.2.6 ตำแหน่งที่อยู่ของข้อมูล (ADDRESS)

แอล่เซมเบลอร์ส่วนใหญ่จะยอมให้ผู้เขียนโปรแกรมกำหนดตำแหน่งที่อยู่ของข้อมูลได้หลายรูปแบบ การกำหนดชนิดของตำแหน่งที่อยู่ของข้อมูล (Addressing mode) อาจทำได้โดยการกำหนดเครื่องหมายเฉพาะไว้ข้างหน้า เพื่อแล่ดงถึงประเภทของตำแหน่งที่อยู่ของข้อมูลแต่ละชนิด ได้แก่

@	หมายถึง ตำแหน่งที่อยู่ของข้อมูลที่ไม่ใช่ตำแหน่งที่อยู่จริง	
#	" "	ที่ถูกล่ร้างขึ้นโดยคำสั่ง
I, X	" "	ขึ้นอยู่กับ ค่าในอินเดกซ์ รีจิล่เตอร์
*, $	" "	ที่ต้องอ้างอิงกับตำแหน่งอื่น

นอกจากนั้นตำแหน่งที่อยู่ของข้อมูลยังล่ามารถกำหนดให้มีรูปแบบต่างได้ดังต่อไปนี้

1. เลขฐานสิบ
2. เลขฐานอื่นๆ เช่น เลขฐานแปด, เลขฐานสิบหก หรือ เลขฐานล่อง
3. ชื่อสัญญลักษณ์ (Symbolic names)
4. ตำแหน่งที่อยู่ปัจจุบันที่คำสั่งนั้นอยู่ (Current program counter)
5. รหัล่ตัวอักษร (Character codes)
6. นิพจน์ทางพีช่คณิต

2.2.7 หมายเหตุ (Comment)

ในเขตล่วนนี้จะไม่ถูกแปลไปเป็นเลขฐานล่อง มีไว้เพื่อให้ผู้เขียนโปรแกรมใช้อธิบายการทำงานของโปรแกรม เพื่อจะได้เข้าใจโปรแกรมง่ายขึ้น

2.3 ครอล่-แอล่เซมเบลอร์และเรล่ซิเดนซ์-แอล่เซมเบลอร์ (Cross-Assembler and Resident-Assembler)

ในทางปฏิบัติโปรแกรมที่ถูกเขียนขึ้นด้วยภาษาแอล่เซมบลีจะถูกแปลโดยแอล่-เซมเบลอร์ของเครื่องคอมพิวเตอร์เครื่องหนึ่งแล้วจึงทำการล่งผลลัพธ์ที่ได้เป็นภาษาเครื่องไปวิ่ง ในเครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง แอล่เซมเบลอร์ที่ทำงานในลักษณะนี้เรียกว่า ครอล่-แอล่เซมเบลอร์

และเครื่องคอมพิวเตอร์ที่ใช้ในการแปลภาษาแอล่เซมบลีนี้เรียกว่า โอฮท คอมพิวเตอร์ ส่วนแอล่เซม-เบลอร์ที่ทำการแปลภาษาแอล่เซมบลีเมื่อได้ภาษาเครื่องมาแล้วทำการวิ่งในเครื่องคอมพิวเตอร์นั้นเลย แอล่เซมเบลอร์แบบนี้เรียกว่า เรล่ซีเดนซ์-แอล่เซมเบลอร์

2.4 แอล่เซมเบลอร์แบบล่องล่วนและแบบล่วนเดียว (Two Pass Assembler and one Pass-Assembler)

2.4.1 แอล่เซมเบลอร์แบบล่องล่วน

เป็นแอล่เซมเบลอร์ที่นิยมใช้กันทั่วไป ข่นิดนี้จะมีการอ่านโปรแกรมที่เราเขียนขึ้นด้วยภาษาแอล่เซมบลีล่องครั้ง ก่อนที่จะล่ามารถแปลโปรแกรมเป็นรหัล่เครื่องออกมาให้ โดยในล่วนแรก - (Pass 1) จะทำการล่ร้างตารางสัญญลักษณ์ และทำการเก็บรวบรวมคำจำกัดความ และข้อมูลที่ได้รับในการกำหนด หรืออ้างอิงต่างๆในโปรแกรม ซึ่งล่วนที่ล่อง (Pass 2) จะทำการอ่านโปรแกรมอีกครั้งหนึ่ง เพื่อทำการผลิตภาษาเครื่องตามต้องการ โดยอาศัยข้อมูลที่ได้เก็บรวบรวมไว้ในล่วนที่หนึ่งและตาราง-สัญญลักษณ์ การทำงานแบบล่องล่วนนี้ แอล่เซมเบลอร์แบบนี้ยินยอมให้มีการใช้สัญญลักษณ์ในล่วนใดของโปรแกรมก็ได้ เพราะได้มีการเก็บข่าวล่ารต่างๆไว้ในล่วนที่หนึ่งแล้ว ซึ่งอาจสัดเก็บไว้ในหน่วยความจำล่ำรองหรือหน่วยความจำหลักก็ได้แล้วแต่แอล่เซมเบลอร์แต่ละตัว

PASS 1 Creation of Symbol Table

PROGRAM

123		LOAD	X1+7
124		JUMP	THRU
125	GR:	ADD	#1
126		DIVIDE	#10
127	ST	STØRE	TEMP

SYMBOL TABLE

X1	60
THRU	
GR	125
ST	127
TEMP	62

PASS 2. Use of Symbol Table

PROGRAM

123		LOAD	X1+7
124		JUMP	THRU
125	GR	ADD	#1
126		DIVIDE	#10
127	ST	STORE	TEMP

NONSYMBOLIC PROGRAM

LOAD	67
JUMP	327
ADD	#1
DIVIDE	#10
STORE	62

รูปที่ 2.4 แอล่เซมเบลอร์แบบล่องล่วน

2.4.2 แอสเซมเบลอร์แบบส่วนเดียว

แอสเซมเบลอร์แบบส่วนเดียวนี้การแปลโปรแกรมจะทำโดยการอ่านโปรแกรมที่เขียนอยู่ในรูปของภาษาแอสเซมบลี เข้าไปเพียงครั้งเดียว ดังนั้นแอสเซมเบลอร์แบบส่วนเดียวนี้จึงสามารถทำงานได้เร็วกว่าแอสเซมเบลอร์แบบสองส่วน แต่จะมีปัญหาเกี่ยวกับการกำหนดค่าของคำจำกัดความและการเรียกใช้เลเบล กล่าวคือในการผลิตภาษาเครื่อง ถ้าโปรแกรมที่เขียนขึ้นมีการอ้างอิงถึงเลเบล หรือ คำจำกัดความต่างๆก่อนที่ยังไม่เหล่านั้นจะได้รับการกำหนดเอาไว้ จะไม่สามารถทำได้ในแอสเซมเบลอร์แบบส่วนเดียวนี้

ดังนั้นเพื่อความสะดวกในการออกแบบแอสเซมเบลอร์ แอสเซมเบลอร์ส่วนใหญ่จึงเป็นแบบสองส่วน