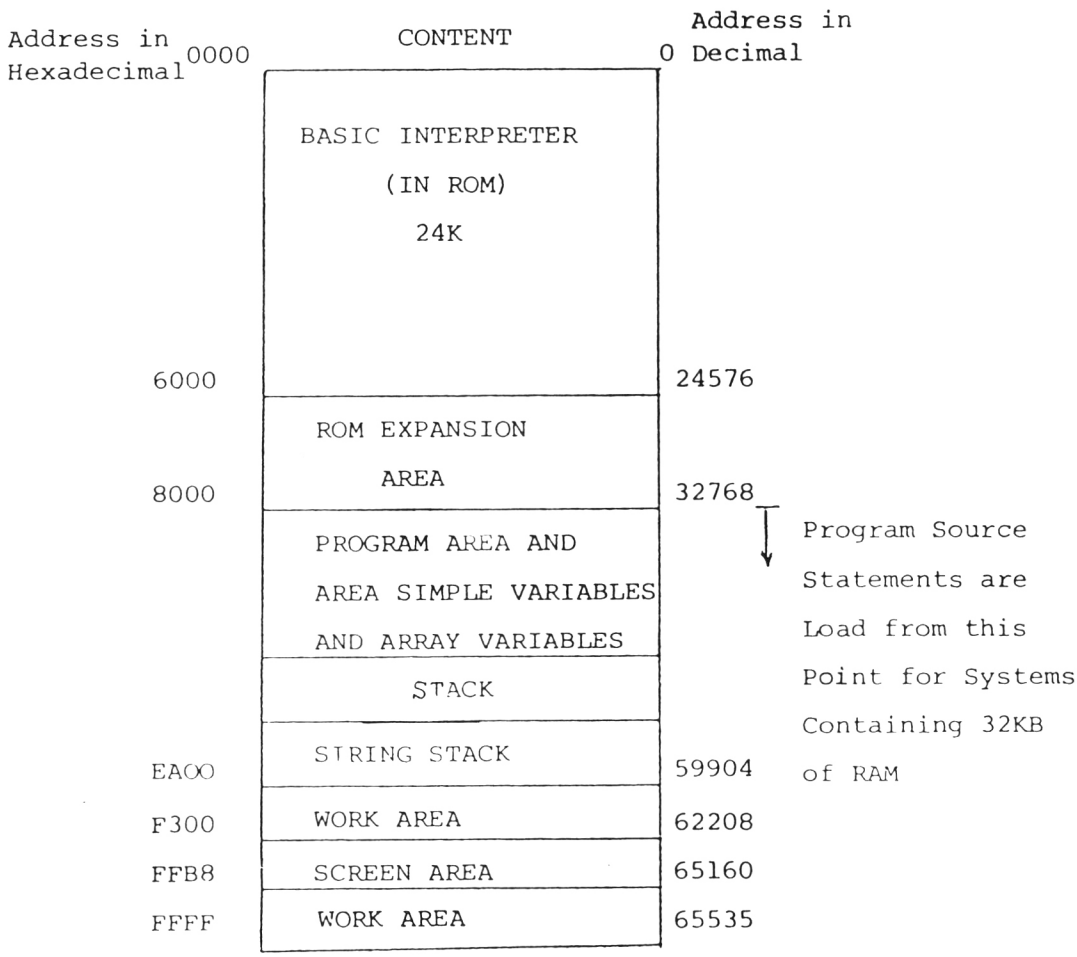
บทที่ 4

การสร้างครอสแอสเซมเบลอร์

4.1 ข้อจำกัดของเครื่องไมโครคอมพิวเตอร์

การวิจัยนี้จะทำการทดลองโดยใช้เครื่องไมโครคอมพิวเตอร์ ของบริษัท NEC รุ่น PC 8000 ซึ่งมีอุปกรณ์ดังนี้

1. หน่วยความจำขนาด 64 กิโลไบต์
2. เครื่องมือป้อนข้อมูล
3. จอภาพ 1 จอ (One List Device)
4. หน่วยขับจานบันทึก (Disk Drive) 2 ตัว
   เครื่องพิมพ์ 1 เครื่อง (Printer)



รูปที่ 4.1 ผังแสดงส่วนความจำ

จากผังแสดงส่วนความจำของเครื่องไมโครคอมพิวเตอร์ จะเหลือส่วนความจำที่ใช้สำหรับการเขียนโปรแกรมด้วยภาษาเบสิกได้ขนาดไม่เกิน 19 กิโลไบต์ ดังนั้นขนาดของตัวโปรแกรมที่จะสร้างขึ้นนี้ได้ต้องไม่เกิน 19 กิโลไบต์ด้วยเช่นเดียวกัน

ในการที่จะเอาตัวแปลโปรแกรมนี้ไปดัดแปลงเพื่อใช้กับเครื่องไมโครคอมพิวเตอร์อื่นนั้นอย่างน้อยจะต้องมีอุปกรณ์ เท่ากับที่ได้กล่าวมาแล้ว ยกเว้น หน่วยขับจานบันทึกมี 1 ตัวก็เพียงพอ

4.2 ข้อกำหนดของครอสแอสเซมเบลอร์

ครอสแอสเซมเบลอร์มีข้อกำหนดของภาษาที่ใช้ดังนี้

4.2.1 รูปแบบของคำสั่งแอสเซมเบลอร์ (Assembler Instruction Format)

เลเบล LABEL	รหัสดำเนินการ OPERATION	โอเพอแรน OPERAND	หมายเหตุ COMMENT

รูปที่ 4.2 คำสั่งของแอสเซมเบลอร์

ประกอบด้วย 4 เขต ดังรูป ซึ่งแต่ละเขต เป็น ฟรี ฟอแม็ท ยกเว้น

1. เลเบล (Label) จะต้องเริ่มต้นที่สดมภ์ ที่ 1
2. ถ้าเป็นบรรทัดหมายเหตุ จะเริ่มต้นด้วย ";" ที่ สดมภ์ ที่ 1
3. แต่ละเขต จะถูกแยกด้วยช่องว่างอย่างน้อย 1 ช่อง

4.2.1.1 เลเบล

เลเบลที่ใช้จะต้องไม่เกิน 6 ตัว และตัวแรกจะต้องเป็นตัวอักษร ส่วนที่เหลือจะเป็นตัวอักษรหรือตัวเลขก็ได้ ยกเว้นห้ามใช้อักษร "x" เป็นเลเบล เพราะเป็นอักษรที่ใช้แทนอินเดกซ์ รีจิสเตอร์

4.2.1.2 รหัสดำเนินการ

คำสั่งของไมโครโปรเซสเซอร์ ที่อยู่ในรูปของนิโมนิค จะถูกใส่ไว้ในส่วนของรหัสดำเนินการ เช่นคำสั่ง INCA เป็นต้น

4.2.1.3 โอเพอแรน

รูปแบบของโอเพอแรน นี้จะขึ้นอยู่กับชนิดของรหัสดำเนินการและการกำหนดตำแหน่งที่อยู่ข้อมูลที่ใช้ เช่น คำสั่ง SWI, RTS เป็นคำสั่งที่ไม่ต้องมีโอเพอแรน เป็นต้น

4.2.1.4 หมายเหตุ

เริ่มต้นด้วยเครื่องหมาย ";"

4.2.2 รูปแบบของการกำหนดตำแหน่งที่อยู่ข้อมูล

มีหลายรูปแบบด้วยกันดังนี้

4.2.2.1 แอ็คคิวมูเลเตอร์ แอดเดรสซิ่ง

การกำหนดตำแหน่งนี้จะไม่มีโอเพอแรน เช่น

```
INCA
```

4.2.2.2 อิมมีเดียท แอดเดรสซิ่ง

โอเพอแรน จะต้องเริ่มต้นด้วยเครื่องหมาย "#" เช่น

```
LDAA    # 0
ADDB    # 1
```

4.2.2.3 ไดเรคท์ แอดเดรสซิ่ง

โอเพอแรน ต้องมีค่าอยู่ระหว่าง 0-255 เช่น

```
LDAA $F2
```

4.2.2.4 เอ็กเท็นเดท แอดเดรสซิ่ง

โอเพอแรน จะเป็นสัญญลักษณ์ หรือค่าก็ได้ และต้องมีค่าอยู่ระหว่าง 0-65535 เช่น

```
LDAB $2A8B
```

4.2.2.5 อินเดกซ์ แอดเดรสซิ่ง

โอเพอแรน ประกอบด้วย 2 ส่วนคือ ส่วนอยู่ก่อนเครื่องหมาย จุลภาคคือ ออฟเซ็ท และหลังเครื่องหมายจุลภาค จะต้องเป็นตัวอักษร"x" เท่านั้น เช่น

```
ADDA 4,X
```

ค่าของออฟเซ็ท จะต้องอยู่ระหว่าง 0-255 และถ้าไม่เขียนส่วนที่เป็นออฟ-เซ็ท พร้อมทั้งเครื่องหมายจุลภาค ก็ได้ โดยค่าออฟเซ็ท ที่ได้จะมีค่าเท่ากับ 0 เช่น

```
ADDA  0,X          จะมีค่าเท่ากับ
ADDA  X
```

4.2.2.6 รีเลทีฟ แอดเดรสซิ่ง

โอเพอแรน เป็นค่ารีเลทีฟกับตัวนับหน่วยความจำ ณ. จุดนั้น เช่น

```
BRA     *+14
```

4.2.3 ข้อกำหนดในการใช้ตัวกำหนดตำแหน่ง (Address Specification)

มีหลายวิธีในการกำหนดโอเพอแรน คือ

4.2.3.1 ชื่อสัญญลักษณ์ (Symbolic Name)

ต้องขึ้นต้นด้วยตัวอักษร และยาวไม่เกิน 6 ตัว เช่น

```
LDAA    TEMP
LDX     #SUM
BRA     LØØP
```

4.2.3.2 ค่าคงที่ที่ใช้เลขฐานสิบหก (Hexadecimal Constant)

ต้องเริ่มต้นด้วยเครื่องหมาย "$" เช่น

```
LDAB    #$FA
DEC     $5A
```

4.2.3.3 ค่าคงที่ที่ใช้เลขฐานแปด (Octal Constant)

ต้องเริ่มต้นด้วยเครื่องหมาย "@" เช่น

```
LDAB    #@13
```

4.2.3.4 ค่าคงที่ที่ใช้เลขฐานสอง (Binary Constant)

ต้องเริ่มต้นด้วยเครื่องหมาย "%" เช่น

```
ADDB    #%10111010
BITA    #%10001111
```

4.2.3.5 ค่าคงที่ที่ใช้เลขฐานสิบ (Decimal Constant)

ไม่ต้องมีเครื่องหมายอะไรนำหน้า เช่น

```
ADDA    15
```

4.2.3.6 ค่าคงที่ที่เป็นตัวอักษรแบบ แอสกี้ 1 ไบต์

ต้องเริ่มต้นด้วยเครื่องหมาย " ' " เช่น

```
LDAA    'B
CMPB    'Z
```

4.2.3.7 นิพจน์ (Expression)

เป็นการประปนกัน ของโอเพอแรนแบบต่างๆกับเครื่องหมาย

+, -, *, / เช่น

```
LDAB    TEMP+1
```

การคำนวณค่าของนิพจน์ จะทำตามหลักพีชคณิต

4.2.3.8 ลิทเทอรอล (Literal)

ประกอบด้วยลิทเทอรอล 3 แบบด้วยกัน

ก) ลิทเทอรอล ที่เป็นฐนิต สองไบต์ จะขึ้นต้นด้วย "=D" เช่น

```
LDAA    =D5
```

ข) ลิทเทอรอล ที่เป็นฐนิต ไบต์เดียว จะขึ้นต้นด้วย "=B" เช่น

```
LDAA    =B5
```

ค) ลิทเทอรอล ที่เป็นค่าคงที่ที่ใช้ตัวอักษร จะขึ้นต้นด้วย "=C" เช่น

```
LDAA    =C'Start'
```

4.2.4 แอสเซมเบลอร์ ไดเรค ทีฟ (Assembler Directive)

ประกอบด้วย ไดเรค ทีฟ ดังนี้

4.2.4.1 ØRG

รูปแบบ : Optional ORG นิพจน์

ไดเรค ทีฟ นี้จะทำการเปลี่ยนค่าของ ตัวที่ใช้นับตำแหน่งที่อยู่ข้อมูล โดยใช้ค่าที่กำหนดในนิพจน์แทน ซึ่งค่านี้จะต้องอยู่ระหว่าง 0000-FFFF (เลขฐานสิบหก) ถ้าค่า ØRG ไม่ถูกกำหนด ตัวที่นับตำแหน่งที่อยู่ข้อมูลจะเริ่มต้นที่ 0

4.2.4.2 END

รูปแบบ : เลเบล END ชื่อสัญญลักษณ์
ใช้แสดงการสิ้นสุดของโปรแกรม

4.2.4.3 EQU

รูปแบบ : เลเบล EQU นิพจน์
ใช้กำหนดค่าให้แก่ ชื่อสัญญลักษณ์ ซึ่งค่าชื่อสัญญลักษณ์นี้จะต้องไม่มีการกำหนดอีกภายหลัง

4.2.4.4 FCB (form Constant Byte)

รูปแบบ: Optional FCB นิพจน์
เป็นการกำหนดค่าใน นิพจน์ เป็นค่ามีความยาว 1 ไบต์ ค่าของ นิพจน์ นี้จะถูกแยกจากกันด้วยเครื่องหมายจุลภาค เช่น

```
L1      FCB     $5, $FF, 32, $17
```

จะถูกจัดเก็บในส่วนความจำดังรูปที่ 4.3

05	FF	20	17

รูปที่ 4.3 แสดงหน่วยความจำ

โดยตำแหน่งของ L1 จะอยู่ที่ตำแหน่งของหน่วยความจำที่เก็บค่า 05

4.2.4.5 FDB (form Double Bytes)

รูปแบบ : Optional FDB นิพจน์
ค่าในนิพจน์ จะถูกจัดเก็บด้วยค่ายาว 2 ไบต์ เช่น

```
L1      FDB     $01F2, $0F
```

จะถูกจัดเก็บในความจำดังรูปที่ 4.4

01	F2	00	0F

รูปที่ 4.4 แสดงหน่วยความจำ

โดยตำแหน่ง ของ L1 จะอยู่ที่ตำแหน่งของหน่วยความจำที่เก็บค่า 01

4.2.4.6 FCC (Form Constants Character)

รูปแบบ : Optional FCC ตัวอักษร
เก็บค่าคงที่ ไว้ในรูปของรหัสแอสกี โดย โอเพอแรน จะต้องเริ่มต้นและปิดท้ายด้วยเครื่องหมาย " ' " เช่น

FCC 'ABC'

จะถูกจัดในส่วนความจำดังรูปที่ 4.5

41	42	43

รูปที่ 4.5 แสดงหน่วยความจำ

4.2.4.7 RMB (Reserve Memory Byte)

รูปแบบ : Optional RMB นิพจน์
จะทำการเปลี่ยนตัวนับตำแหน่งที่อยู่ข้อมูล ไปเป็นค่าเท่ากับ ค่าในตัวนับตำแหน่งที่อยู่ข้อมูล บวกกับค่าใน นิพจน์ ซึ่งมีผลให้ตำแหน่งที่อยู่ข้อมูลบางส่วนในหน่วยความจำถูกสำรองไว้

4.2.4.8 LØRG

รูปแบบ : LØRG
เป็นตัวบอกให้กำหนดค่าตำแหน่งที่อยู่ของลิทเทอรอล ที่ใช้เมื่อพบคำสั่งนี้

4.3 การออกแบบครอส-แอสเซมเบลอร์

ในการออกแบบครอส-แอสเซมเบลอร์นี้ เพื่อที่จะให้เราสามารถอ้างถึงสัญลักษณ์ไว้ที่ใดในโปรแกรมก็ได้ จึงเลือกใช้การทำงานแบบ 2 ส่วน โดยแต่ละส่วน มีวัตถุประสงค์และการทำงานดังนี้

ส่วนที่ 1: วัตถุประสงค์เพื่อสร้างตารางสัญญลักษณ์ (Symbol Table) และให้ค่ารหัสเครื่องแทนนีโมนิค ที่ใช้ในคำสั่ง ตลอดจนค่าตำแหน่งที่อยู่ข้อมูลที่ไม่ได้กำหนดเป็นสัญญลักษณ์ มีการทำงานดังนี้

1. สร้างตารางสัญญลักษณ์
2. หา รหัสเครื่อง โดยหาจากตารางออพโคด (Opcode Table) และตารางเครื่อง (Machine Table)
3. คำนวณค่า ตัวนับตำแหน่งที่อยู่ข้อมูล
4. ทำงานตามแอสเซมเบลอร์ไดเรค ทีฟ

ส่วนที่ 2 : วัตถุประสงค์ เพื่อให้ค่ารหัสเครื่อง ในส่วนที่เหลือจากส่วนที่ 1

1. หาสัญญลักษณ์ ที่ยังไม่ได้ให้ค่ารหัสเครื่อง
2. ให้ค่าสัญญลักษณ์ โดยการค้นจาก ตารางสัญญลักษณ์

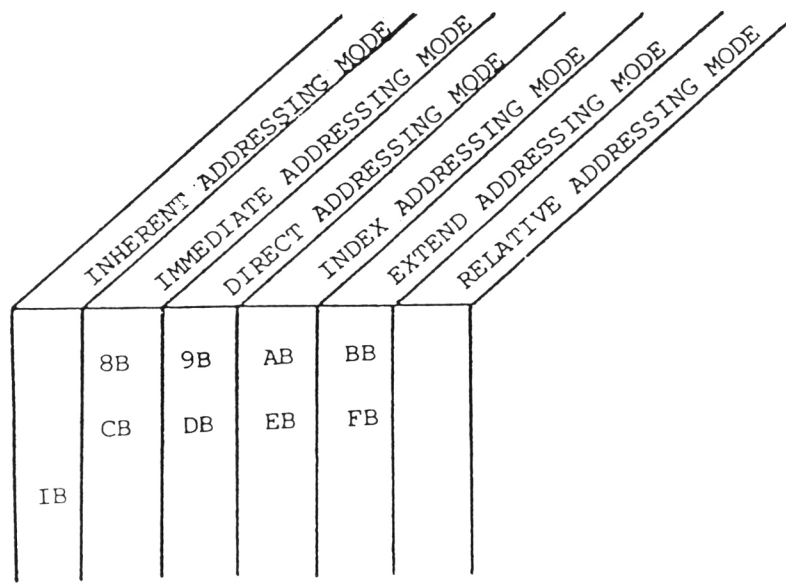
4.3.1 โครงสร้างข้อมูล

หัวข้อนี้จะกล่าวถึงข้อมูลและ ตารางต่างๆที่ใช้

1. โปรแกรมข้อมูลเข้า คือ โปรแกรมแอสเซมบลีที่เขียนขึ้นตามข้อกำหนดในข้อ-4.2 โดยเขียนใน โมด(Mode) ของเบสิก แล้วจัดเก็บไว้ในแฟ้มข้อมูลในลักษณะของแฟ้มข้อมูลแบบ แอสกี้ เบสิก
2. ตารางออพโคด เป็นตารางที่จัดเก็บรหัสนีโมนิค โดยการเก็บเรียงกันไปตามลำดับที่อยู่ในภาคผนวก ดังตัวอย่าง

   "ADDA", "ADDB", "ABA", "ADCA"

3. ตารางเครื่อง ใช้เก็บค่าเลขฐานสิบหก ที่ตรงกับค่ารหัสนีโมนิค โดยแบ่งตามการกำหนดตำแหน่งที่อยู่ข้อมูล แบ่งออกเป็น 6 แบบดังรูปที่ 4.6



รูปที่ 4.6 รูปแบบของตารางเครื่อง

คำสั่งที่ไม่มีการกำหนดตำแหน่งที่อยู่ข้อมูล แบบใดช่องนั้นจะแทนด้วย Blank ไว้

4. ตารางสัญญลักษณ์ ใช้เก็บชื่อสัญญลักษณ์, ค่า ของชื่อสัญญลักษณ์ และเลขที่บรรทัด ที่ใช้ ดังนี้

Symbol	Value	Line No
L1	5	20
AB	70	30

รูปที่ 4.7 ตารางสัญญลักษณ์

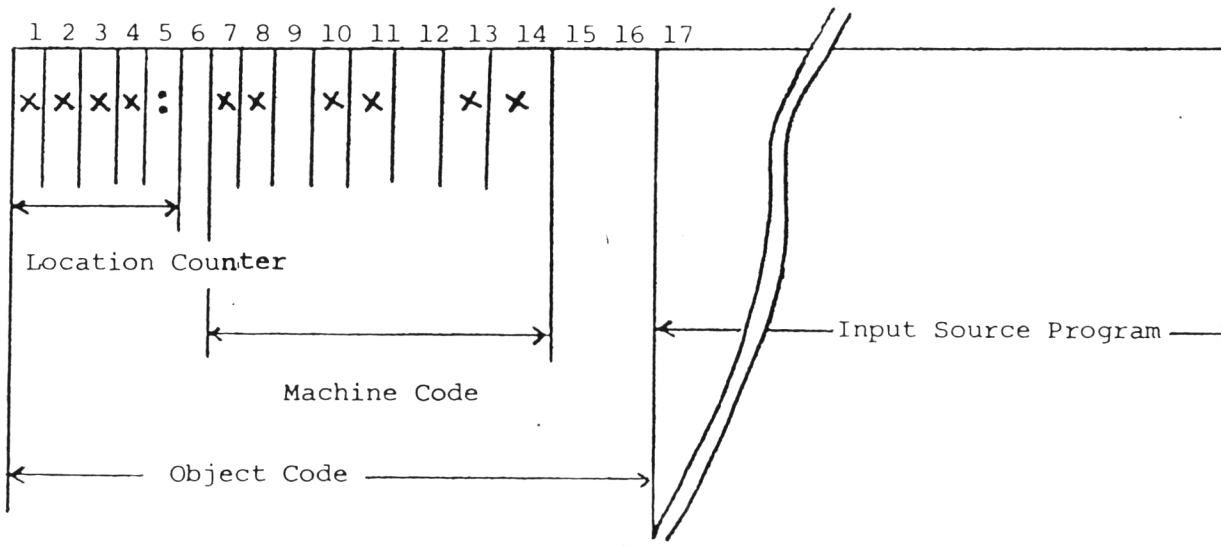
5. รูปแบบของข้อมูลออก แบ่งออกเป็น 2 ส่วนคือ

1. รหัสเครื่อง ที่เครื่องแปลได้ ประกอบด้วย

1.1 ตัวนับตำแหน่งที่อยู่ข้อมูล

1.2 รหัสเครื่อง

2. โปรแกรมข้อมูลเข้า



รูปที่ 4.8 ผลลัพธ์ของแอสเซมเบลอร์

4.4 ข่าวสารของความผิดพลาด (Error Message)

แอสเซมเบลอร์จะทำการเตรียมข่าวสารของความผิดพลาด ดังแสดงไว้ข้างล่างนี้ โดยจะทำการพิมพ์ข่าวสารนั้นหลังจากบรรทัดที่เกิดความผิดพลาดนั้น เพื่อว่าผู้เขียนโปรแกรมจะได้สามารถทำการตรวจสอบและทำการแก้ไขได้สะดวกยิ่งขึ้น

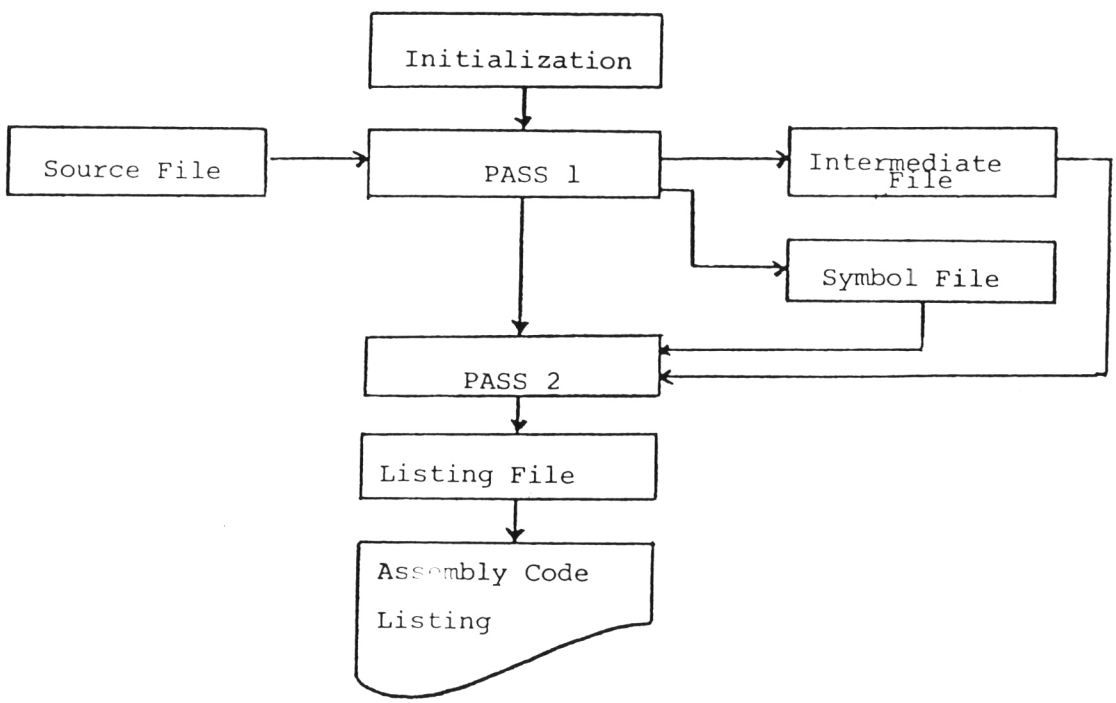
ข่าวสารของความผิดพลาด	ความหมาย
1. Unknown Opcode	- รหัสดำเนินการ ที่ใช้ในคำสั่งไม่มีใช้ในภาษาแอสเซมบลีของไมโครโปรเซสเซอร์เบอร์ 6800
2. Missing Opcode	- คำสั่งไม่มี รหัสดำเนินการ
3. Missing Operand	- คำสั่งที่ควรจะมี โอเพอแรน แต่ไม่มีการกำหนดไว้ในคำสั่ง
4. Invalid Addressing	- ใช้คำสั่งผิดรูปแบบของการกำหนดตำแหน่งที่อยู่ข้อมูล
5. Branch Out of Range	- ระยะทางที่ Branch ไปเกิน+127 และ-128
6. Invalid Number	- ตัวอักษรหรือตัวเลขที่ใช้ไม่มีในระบบตัวเลขที่ใช้
7. Invalid Delimiter	- ใช้ตัวแบ่งเขตข้อมูลผิดชนิด
8. Invalid Label in Memory Assignment	- มีการอ้างถึงชื่อที่ไม่ได้กำหนดไว้
9. Invalid Literal	- ใช้รูปแบบของลิทเทอรอลผิด
10. Multiple Definition	- มีการกำหนดชื่อซ้ำซ้อนกัน
11. Value too Large	- ค่าที่ใช้เกินค่าที่บังคับให้ใช้ได้
12. Irresolve fwd ref Bad Label	- ค้นหาชื่อที่กำหนดไว้ในตารางสัญญลักษณ์ไม่พบ

ตารางที่ 4.1 ข่าวสารของความผิดพลาด

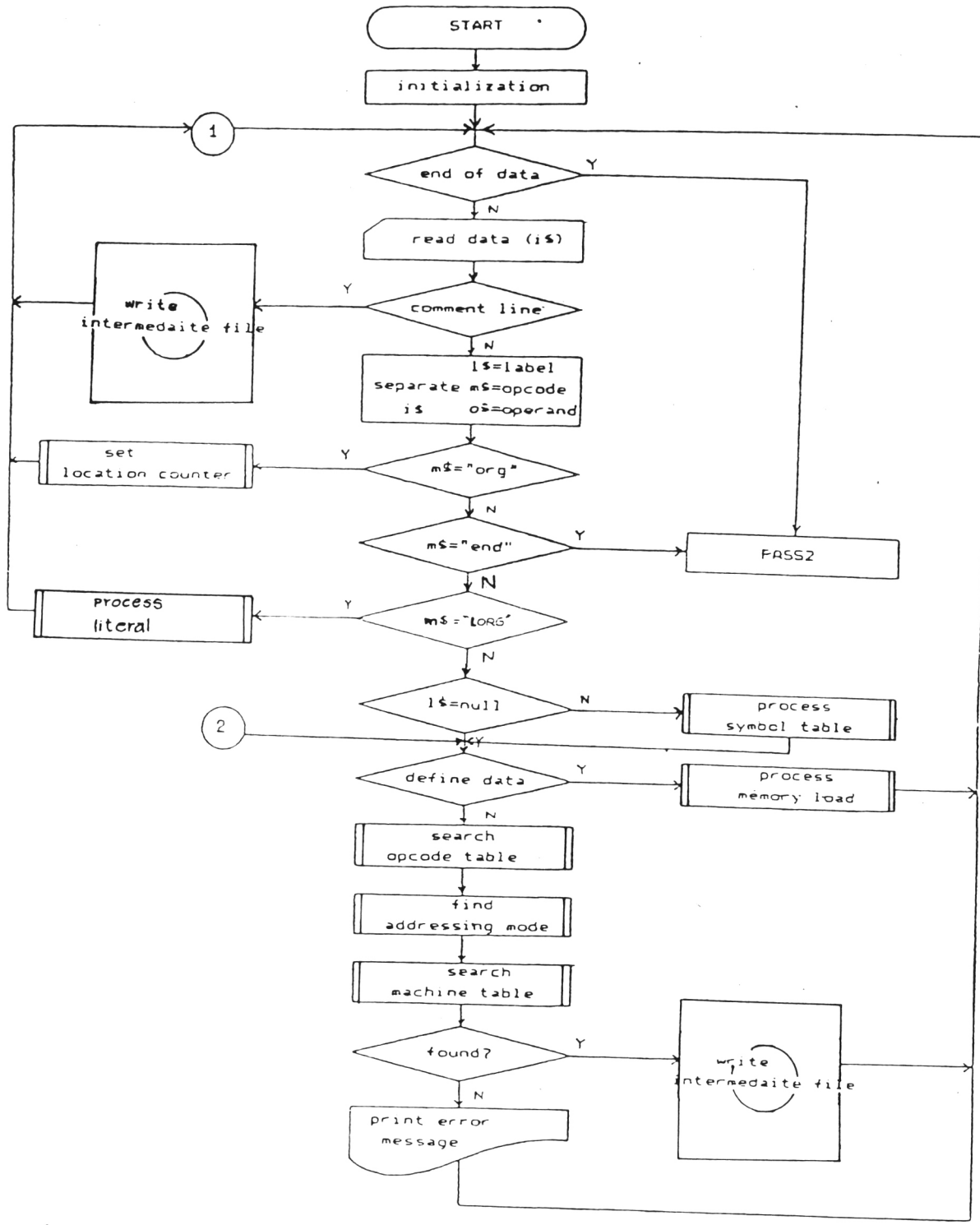
4.5 ขั้นตอน (Algorithm)

หัวข้อนี้จะแสดงถึงผังงาน การทำงานของโปรแกรมแอสเซมเบลอร์

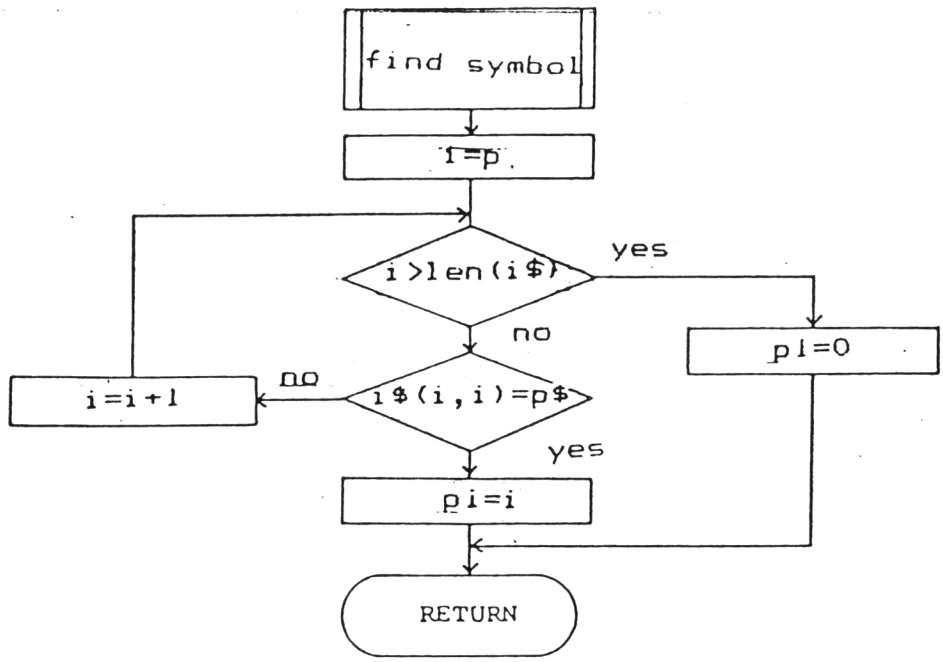
เนื่องจากรหัสเครื่อง ที่ตรงกับ รหัสนีโมนิค นั้นขึ้นอยู่กับการกำหนดตำแหน่งที่อยู่ข้อมูล ดังนั้นเพื่อไม่ให้เป็นการเสียเวลาที่ต้องใช้ในการตรวจหาเพื่อให้ทราบว่า รหัสนีโมนิคใด มีการกำหนดตำแหน่งที่อยู่ข้อมูล แบบใด และมีค่ารหัสเครื่อง เท่าใด ถึงสองครั้งคือในส่วนที่ 1 และส่วนที่2 จึงได้มีการให้ค่ารหัสเครื่อง ตั้งแต่ในส่วนที่ 1 เลย แล้วทำงานเพิ่มค่าของตัวนับตำแหน่งที่อยู่ข้อมูลตามความยาวของคำสั่งนั้น และค่าโอเพอแรน ที่แปลได้ ก็จะแปลเลย ในส่วนที่ 1 นี้ ส่วนที่เหลือจึงจะทำในส่วนที่ 2



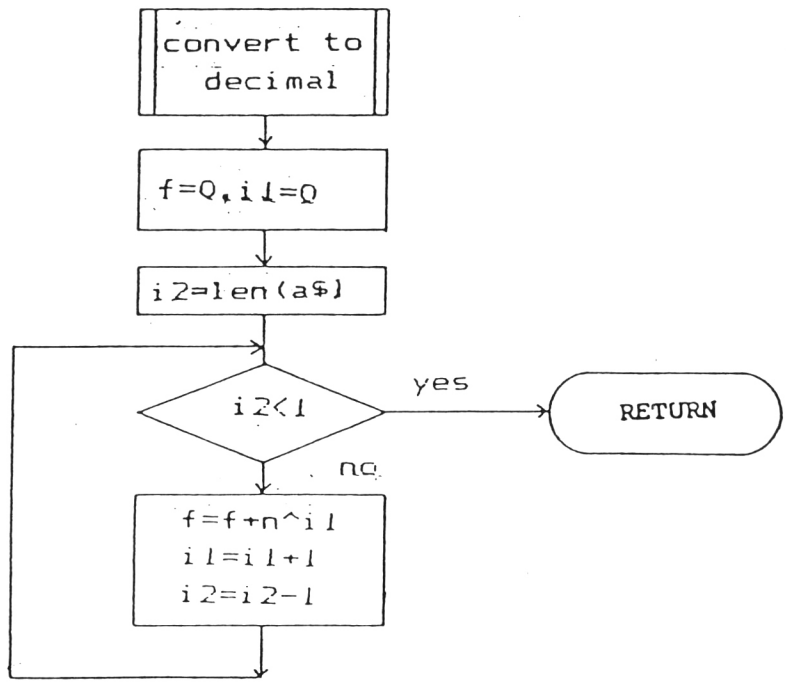
รูปที่ 4.9 ผังแสดงการทำงานของครอส-แอสเซมเบลอร์



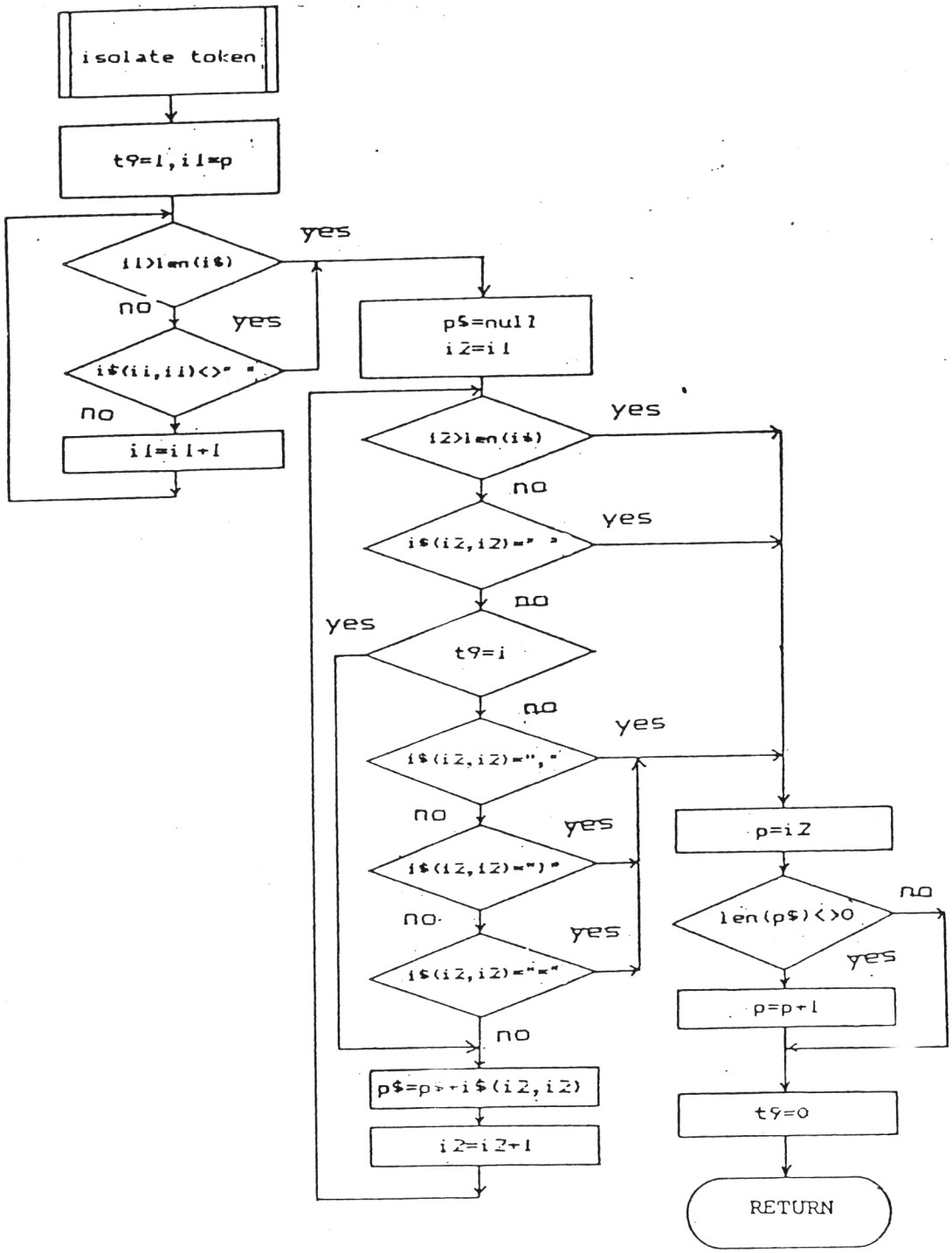
รูปที่ 4.10 ขั้นตอนการทำงานของแอสเซมเบลอร์ส่วนที่ 1



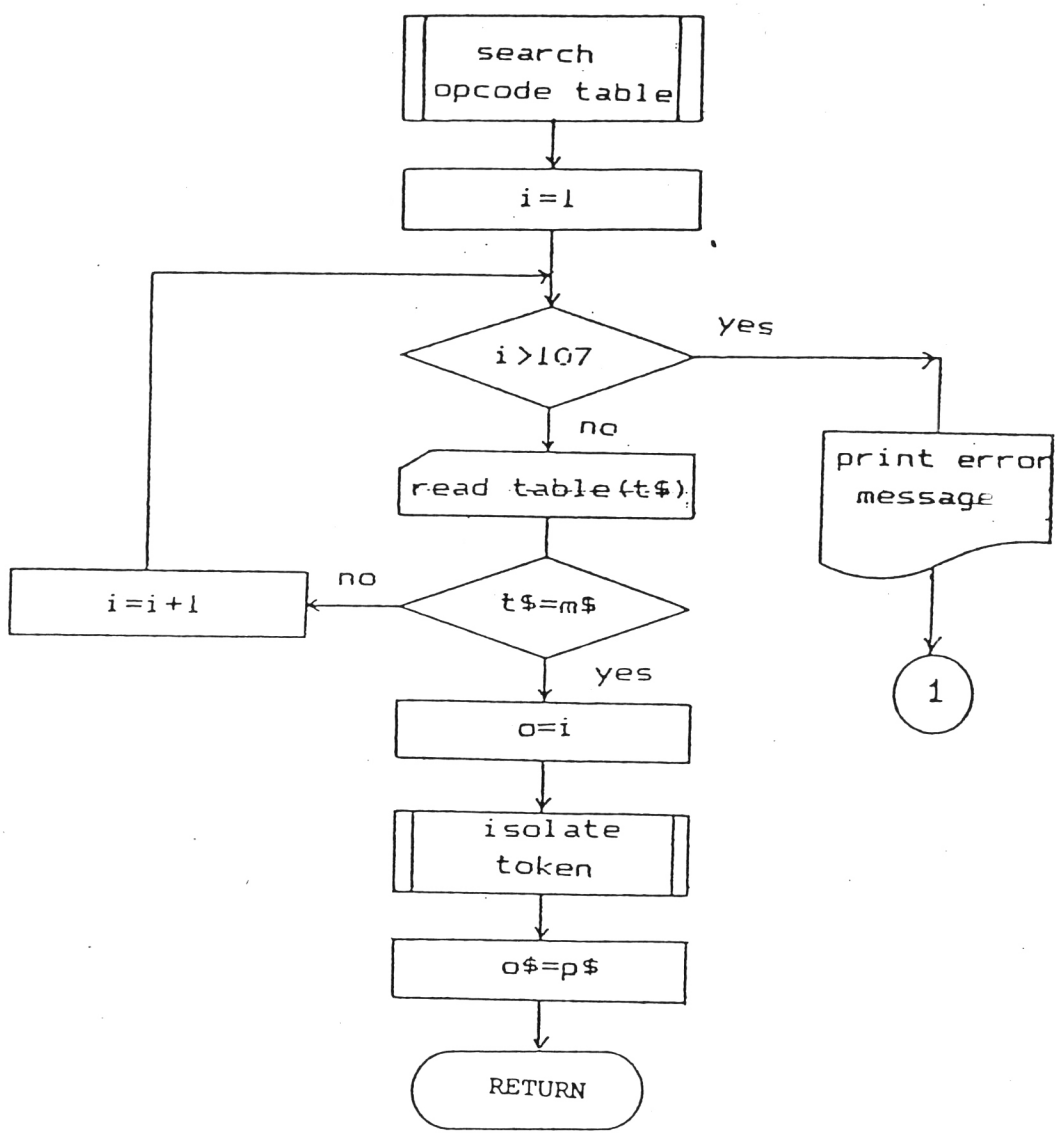
รูปที่ 4.11 ขั้นตอนการทำงานของโปรแกรมย่อย Find Symbol



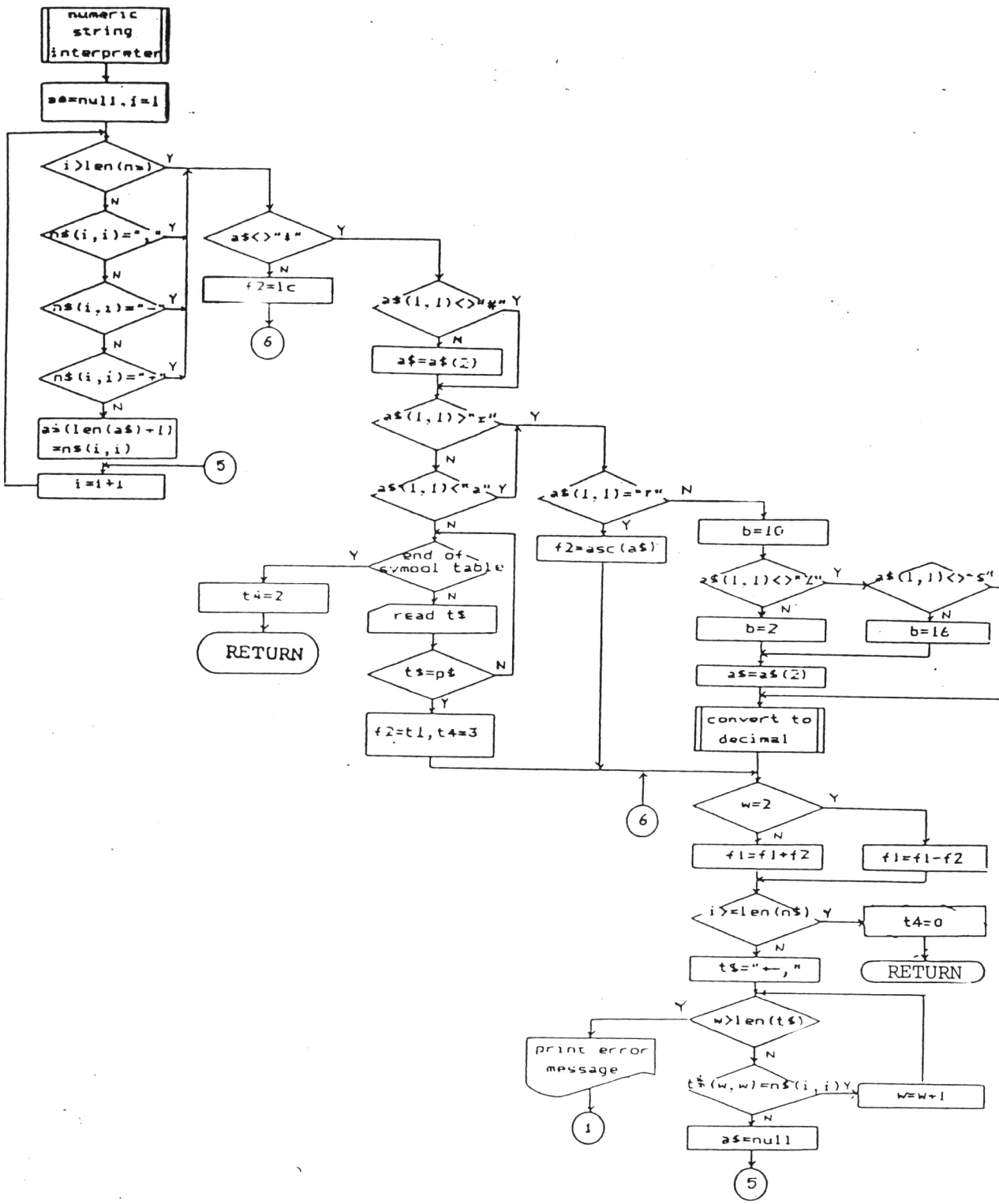
รูปที่ 4.12 ขั้นตอนการทำงานของโปรแกรมย่อย Convert to Decimal



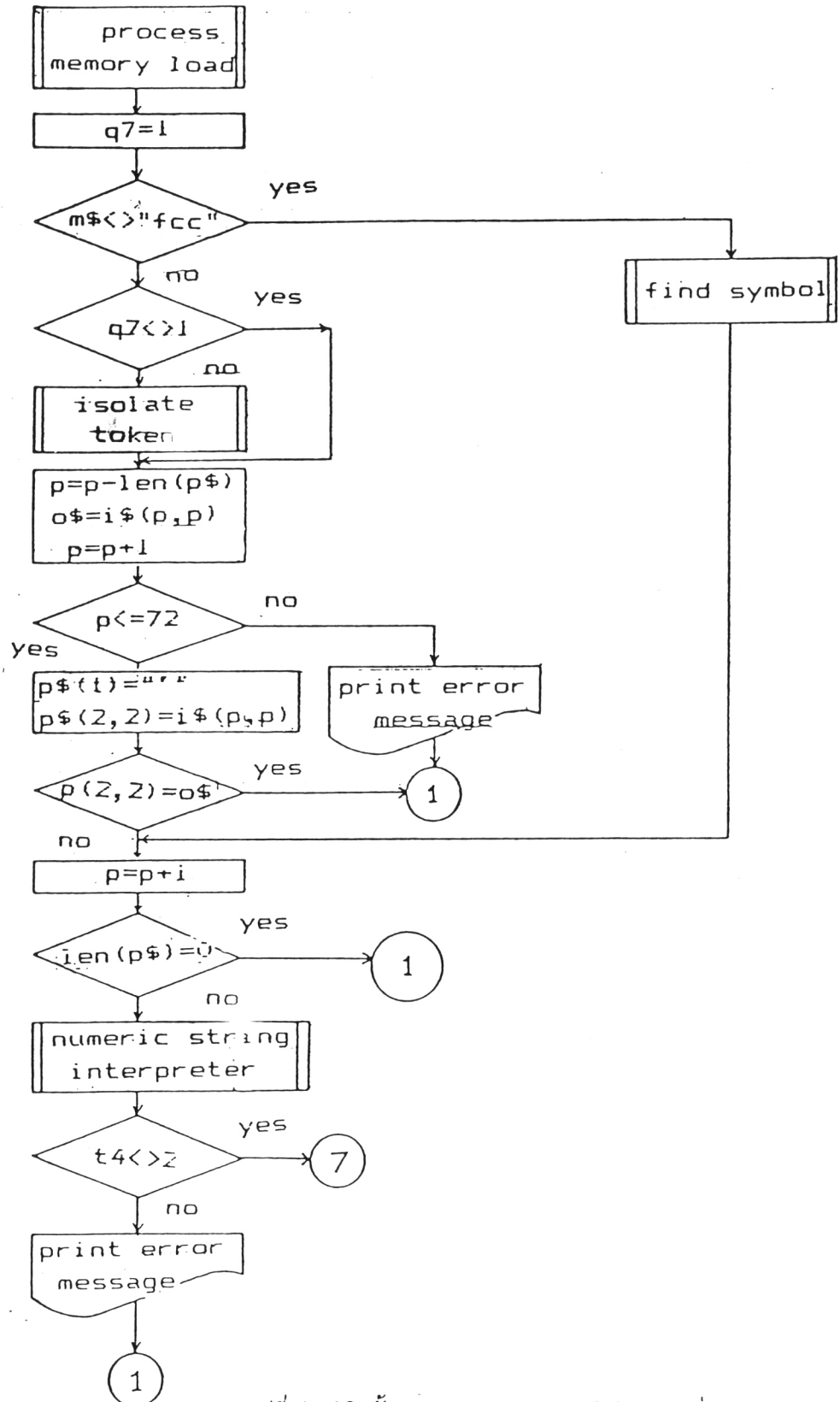
รูปที่ 4.13 ขั้นตอนการทำงานของโปรแกรมย่อย Isolate Token



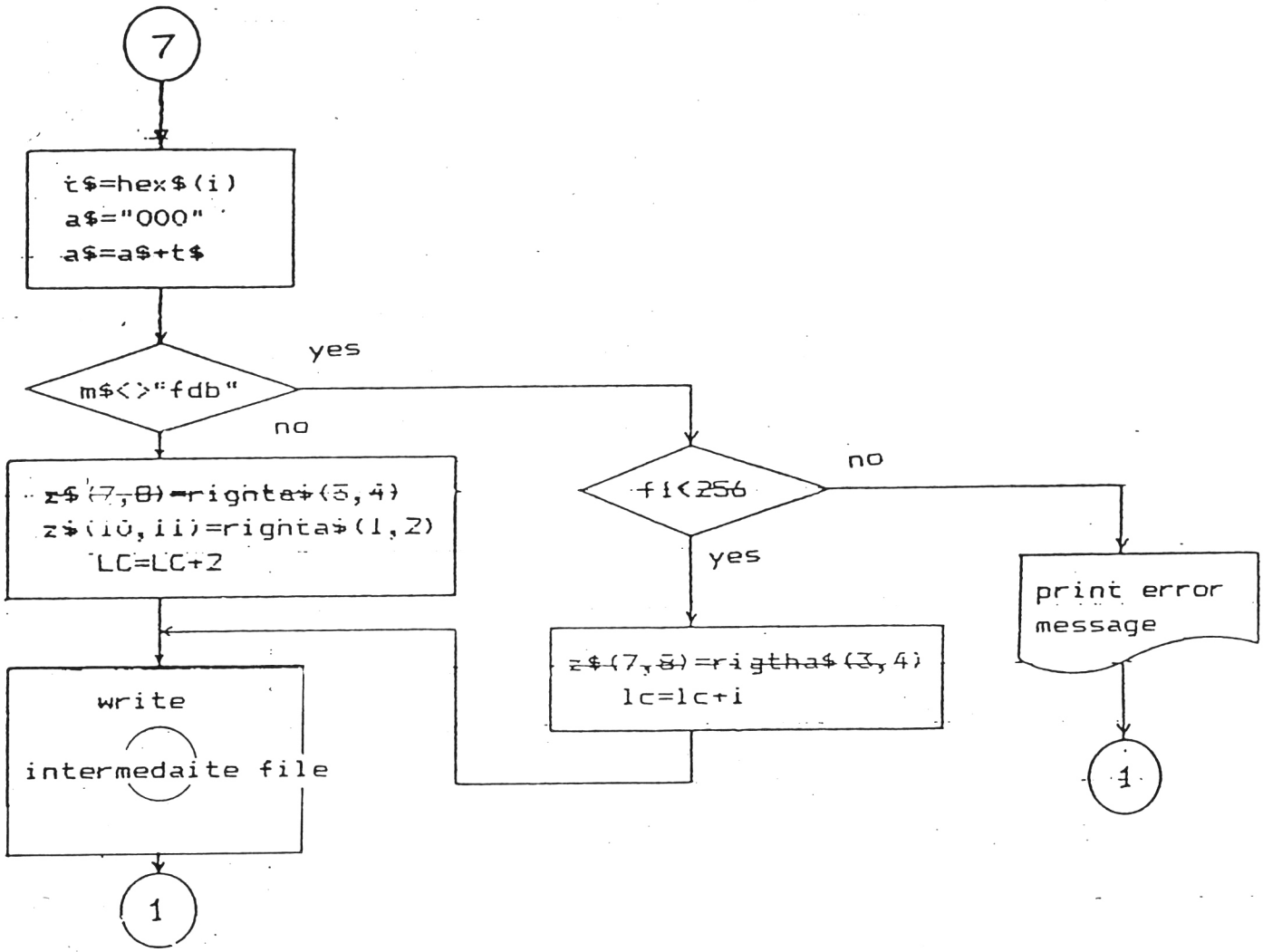
รูปที่ 4.14 ขั้นตอนการทำงานของโปรแกรมย่อย Search Opcode Table



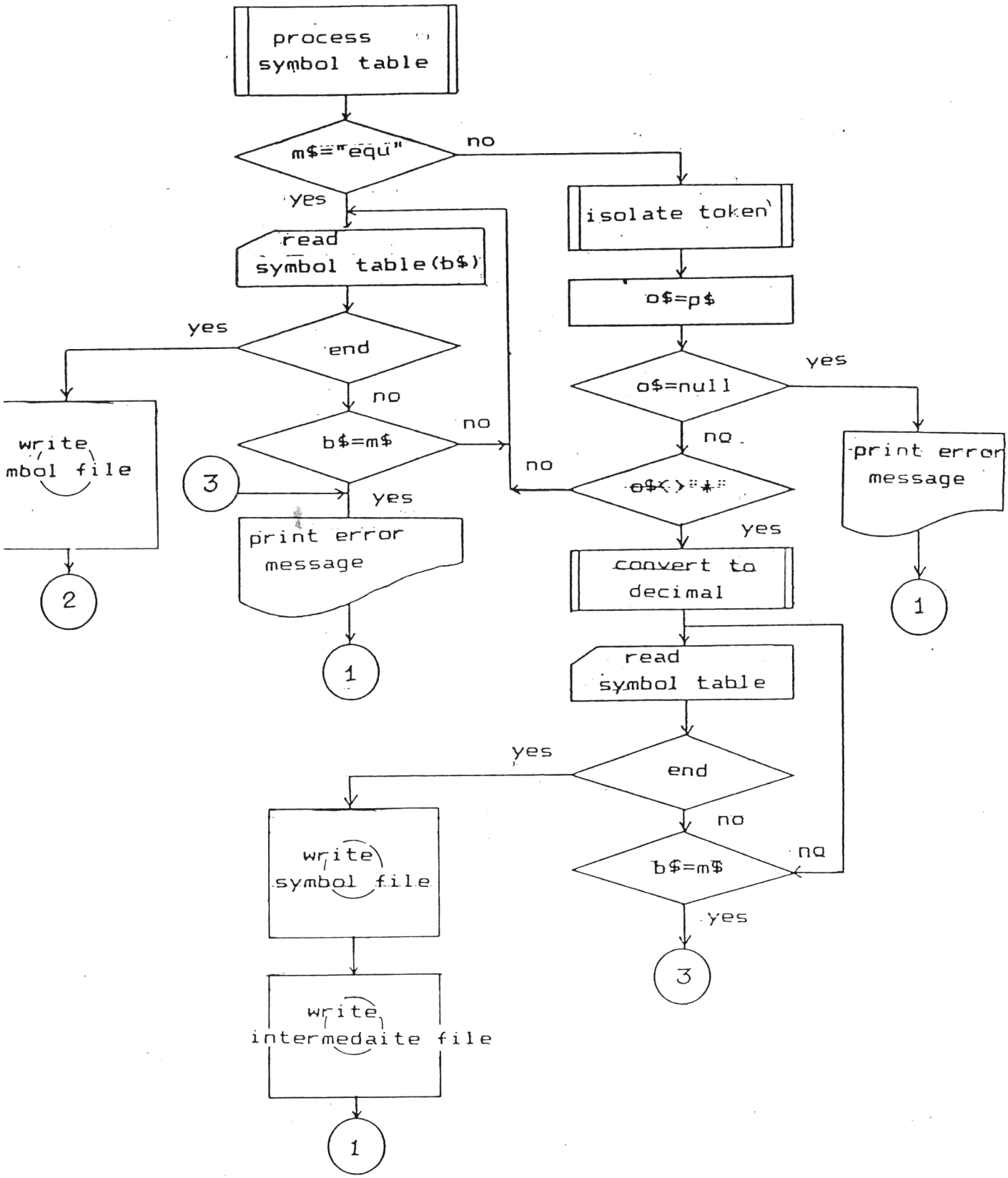
รูปที่ 4.15 ขั้นตอนการทำงานของโปรแกรมย่อย Numeric String Interpreter



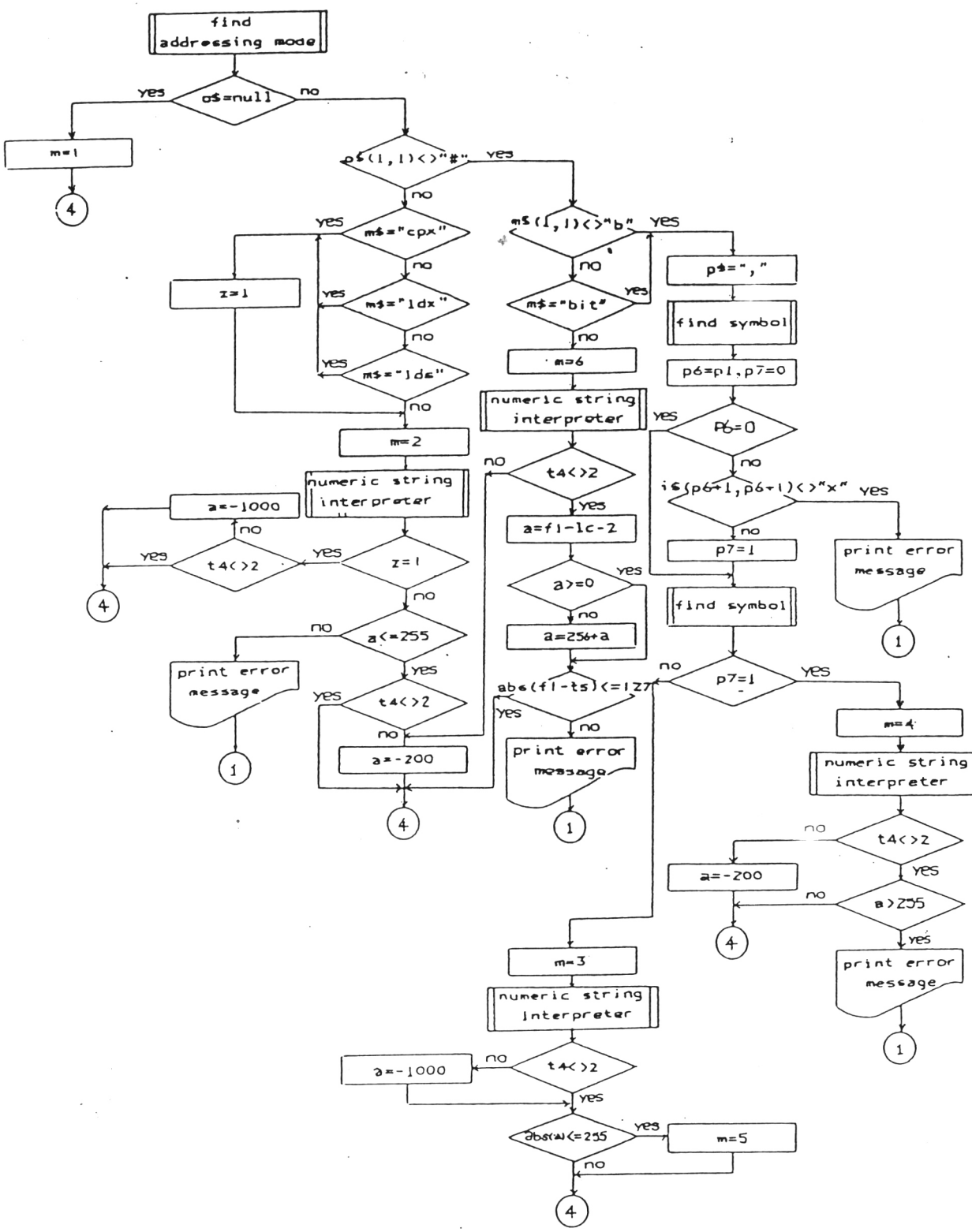
รูปที่ 4.16 ขั้นตอนการทำงานของโปรแกรมย่อย Process Memory Load



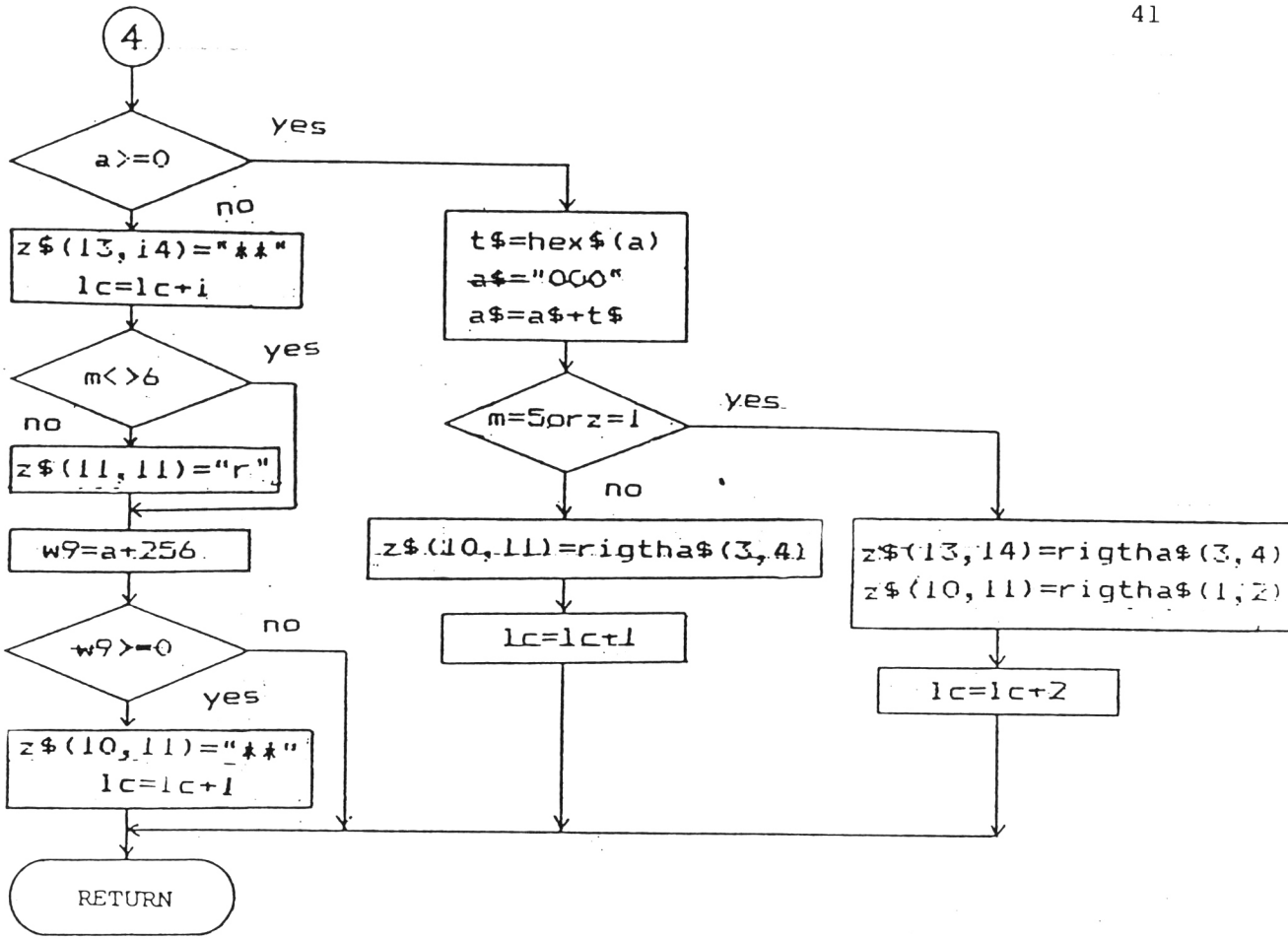
รูปที่ 4.17 ขั้นตอนการทำงานของโปรแกรมย่อย Process Memory Load(ต่อ)



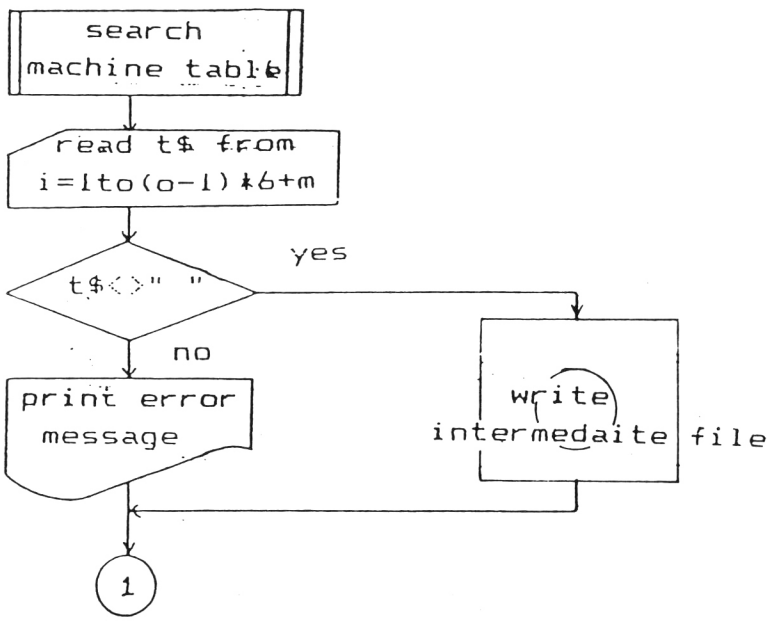
รูปที่ 4.18 ขั้นตอนการทำงานของโปรแกรมย่อย Process Symbol Table



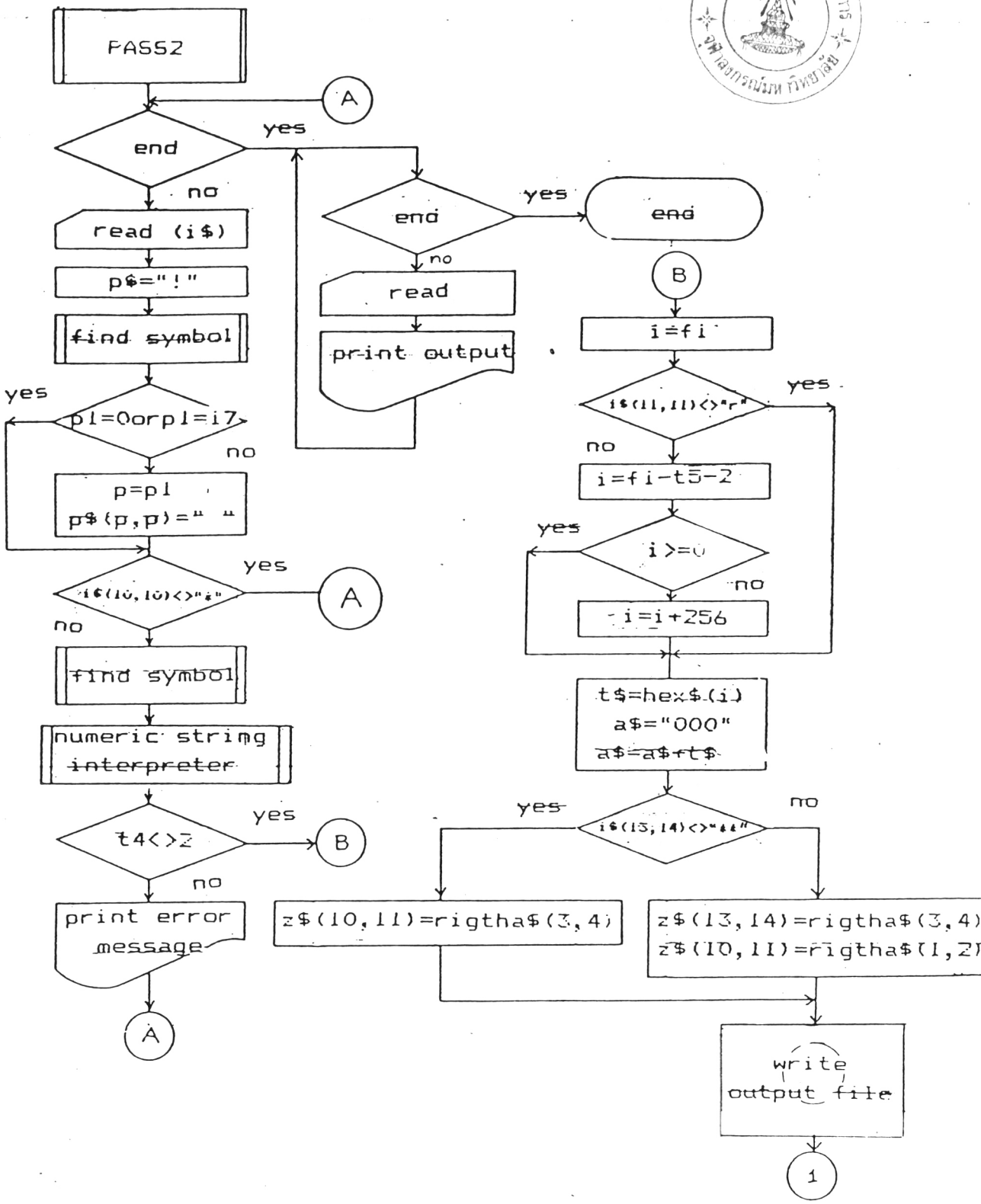
รูปที่ 4.19 ขั้นตอนการทำงานของโปรแกรมย่อย Find Addressing Mode



รูปที่ 4.20ขั้นตอนการทำงานของโปรแกรมย่อย Find Addressing Mode (ต่อ)



รูปที่ 4.21 ขั้นตอนการทำงานของโปรแกรมย่อย Search Machine Table



รูปที่ 4.22 ขั้นตอนการทำงานของแอสเซมเบลอร์ส่วนที่ 2

4.6 การใช้ครอส-แอสเซมเบลอร์

ครอสแอสเซมเบลอร์ที่ทำการสร้างขึ้นนี้จะถูกสั่งเก็บไว้ในแผ่นจานแม่เหล็กโดยใช้ "ASSM68" และโปรแกรมที่จะทำการออกผลลัพธ์ในลักษณะที่มีเฉพาะตำแหน่ง กับรหัสเครื่อง เท่านั้น ชื่อ "LIST" สำหรับการใช้โดยละเอียดมีดังนี้

1. เปิดเครื่องคอมพิวเตอร์ซึ่งจะอยู่ในโมดของเบสิก แล้วทำการเขียนโปรแกรม

```
100 START       ORG     100
110             LDAA    #$7
120             STAA    TEMP+5*2
125 SOUPP       EQU     7
130             LDAA    =D$40,5,7
140             LORG
150             STAA    TEMP+1
160             LDX     =C'start'
170             LDAA    0,X
180             STAA    *+1
190             SWI
200             FCB     0,1,4,88
210 DEF         RMB     7
220 SOTMU       EQU     9
230 SOTBL       EQU     6
240             LORG
250 TEMP        RMB     5
260             END     START
```

รูปที่ 4.23 ตัวอย่างโปรแกรมภาษาแอสเซมบลี

2. ทำการสั่งเก็บโปรแกรมที่เขียนขึ้นไว้ในแผ่นจานแม่เหล็ก ในลักษณะของแอสกี้ โดยใช้ SAVE "Program Name", a

3. โหลดแอสเซมเบลอร์โปรแกรมเข้ามาโดยใช้คำสั่ง

   LOAD "ASSM68"

4. กดปุ่ม RUN ซึ่งจะมีคำถามทางจอภาพ (CRT) ดังนี้

   1) NAME OF PROGRAM FILE?
      ตอบชื่อโปรแกรมที่ทำการ SAVE ไว้แล้วกด RETURN

   2) NAME OF OBJECT FILE YOU WANT SAVE?
      ตอบชื่อที่ต้องการ เช่น "0" แล้วกด RETURN

   3) DO YOU WANT LISTING ? (Y/N)

ตอบ Y ถ้าต้องการผลลัพธ์ทางเครื่องพิมพ์

ตอบ N จะได้ผลลัพธ์ออกทางจอภาพ

หลังจากนี้จะปรากฏที่จอภาพ ว่า

ASSEMBLER BEGIN

เป็นอันว่าโปรแกรมแอสเซมเบลอร์เริ่มทำงาน ซึ่งจะได้ ผลลัพธ์ออกดังแสดงในรูปที่ 4.24 และสำหรับในกรณีที่ต้องการใช้ภายหลัง หรือในลักษณะที่มีเฉพาะตำแหน่งที่อยู่กับค่ารหัสเครื่องเท่านั้น ก็ทำได้โดยการโหลด โปรแกรม "LIST" เข้ามาแล้วกดปุ่ม RUN ซึ่งจะปรากฏคำถามทางจอภาพ ดังนี้

"WHAT IS YOUR OBJECT FILE NAME"?

ตอบชื่อ Object File แล้วกด RETURN

DØ YOU WANT LISTING OR MEMORY DUMP ?

ENTRY 1 FØR LISTING, 2 FØR MEMORY DUMP

ซึ่งถ้าตอบ 1 จะได้ข้อมูลออกดังรูปที่ 4.24 และถ้าตอบ 2 จะได้ดังรูปที่ 4.25

4.7 การนำผลลัพธ์จากการแอสเซมบลี ไปใช้กับแผ่นพิมพ์เดี่ยว

ในการป้อนข้อมูลเข้าเครื่องแผ่นพิมพ์เดี่ยว กระทำได้โดยการป้อนข้อมูลผ่านทางเครื่องมือป้อนข้อมูล โดยเริ่มจากการตั้งตัวนับหน่วยความจำให้ตรงกับตำแหน่งของ ØRG ที่แปลได้ แล้วทำการป้อนรหัสเครื่องที่แปล ได้ซึ่งอยู่หลังจากเครื่องหมาย ":" ในลักษณะของเลขฐานสิบหกทีละตัว โดยตัวนับหน่วยความจำของเครื่องจะเพิ่มขึ้นละหนึ่ง ทำการป้อนรหัสเครื่องที่ได้จนหมดโปรแกรม

สำหรับการที่จะสั่งให้โปรแกรมเริ่มวิ่งที่จุดใดนั้นดูได้จากตัวนับหน่วยความจำที่ตรงกับคำสั่ง END แล้วตั้งตัวนับหน่วยความจำของเครื่องให้ตรงกับตัวเลขที่แปลได้นี้แล้วจึงกดคำสั่งให้วิ่งโปรแกรมได้

```
0064:                  100 START       ORG     100
0064: 86 07            110             LDAA    #$7
0066: B7 00 77         120             STAA    TEMP+5*2
0069:                  125 SQUPP       EQU     7
0069: B6 00 6C         130             LDAA    =D$40,5,7
006C:                  140             LORG
006C: 00 40                  =D$40,5,7
006E: 00 05
0070: 00 07
0072: B7 00 8E         150             STAA    TEMP+1
0075: FE 00 88         160             LDX     =C'start'
0078: A6 00            170             LDAA    0,X
007A: 97 7B            180             STAA    *+1
007C: 3F               190             SWI
007D: 00               200             FCB     0,1,4,88
007E: 01
007F: 04
0080: 58
0081:                  210 DEP         RMB     7
0088:                  220 SQTBU       EQU     9
0088:                  230 SQTBL       EQU     6
0088:                  240             LORG
0088: 73                     =C'start'
0089: 74
008A: 61
008B: 72
008C: 74
008D:                  250 TEMP        RMB     5
0064:                  260             END     START
SYMBOL   TABLE  6800       line#
START           0064       100
SQUPP           0007       125
=D$40,5,7       006C       140     130
DEP             0081       210
SQTBU           0009       220
SQTBL           0006       230
=C'start'       0088       240     160
TEMP            008D       250     120     150
TOTAL ERROR  0
MEMORY SIZE = 46    BYTE
```

รูปที่ 4.24 ผลลัพธ์ของแอสเซมเบลอร์

```
0064: 86 07 B7 00 97 B6 00 8C 00 40 00 05 00 07 B7 00
0074: 8E FE 00 88 A6 00 97 78 3F 00 01 04 53
0084:                73 74 61 72 74
```

รูปที่ 4.25 แสดงผลลัพธ์ในลักษณะที่มีเฉพาะรหัสเครื่อง