บทที่ 7

แนวความคิดในการพัฒนาโปรแกรมจริง

7.1 การแบ่งโปรแกรมออกเป็นส่วน ๆ

โปรแกรมที่พัฒนาขึ้น จะแบ่งออกเป็น 3 ส่วน แยกออกจากกัน คือ

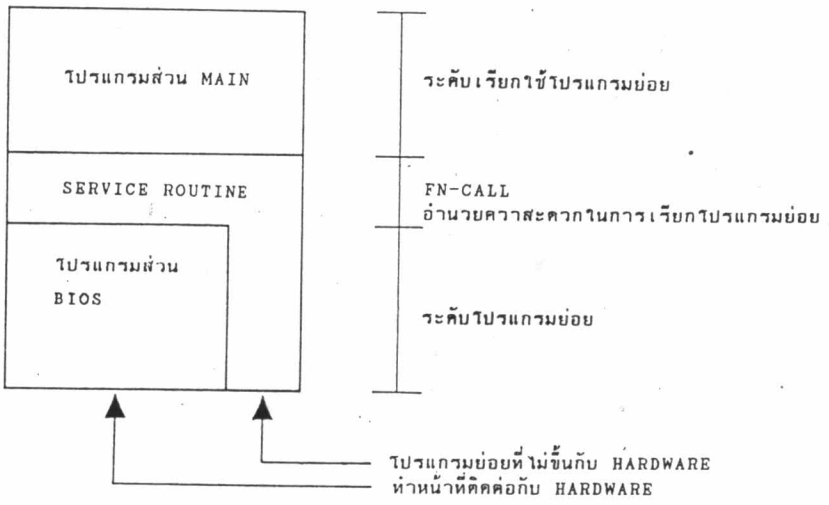
1. ส่วนโปรแกรมหลัก (MAIN)
2. ส่วนบริการโปรแกรมย่อย (SR)
3. ส่วนโปรแกรมควบคุม อินพุท เอ้าท์พุท (BIOS)

ในส่วนของโปรแกรมหลัก (MAIN) เป็นส่วนที่ควบคุมการทำงานแบบสถานะ ของตู้สาขาโทรศัพท์ ประกอบด้วย โปรแกรมประจำสถานะต่าง ๆ ซึ่งทำการแปลงมาจากภาษา STL โดยการแปลงคำสั่งในภาษา STL มาเป็นการเรียกใช้โปรแกรมย่อย ซึ่งอยู่ในส่วนบริการโปรแกรมย่อย และโปรแกรมควบคุมอินพุท เอ้าท์พุท สาเหตุที่แยกโปรแกรมหลักออกมาเป็นส่วนหนึ่ง ก็เนื่องจากโปรแกรมส่วนนี้ ขึ้นอยู่กับบริการพิเศษของตู้สาขาโทรศัพท์ การแยกออกมาต่างหากทำให้การพัฒนาการให้บริการพิเศษ สามารถแยกทำจากโปรแกรมส่วนอื่น ๆ

ส่วนบริการโปรแกรมย่อย (SERVICE ROUTINE หรือเรียกย่อว่า SR) และส่วนโปรแกรมควบคุมอินพุท เอ้าท์พุท (BIOS) เป็นส่วนที่ประกอบด้วยโปรแกรมย่อยจำนวนหนึ่งให้โปรแกรมหลักเรียกใช้ โปรแกรมย่อยเหล่านี้ก็คือ คำสั่งในภาษา STL การเรียกใช้ โปรแกรมหลักจะส่งหมายเลขโปรแกรมย่อยที่ต้องการมาทำงาน โดยกำหนดให้จุดเริ่มต้นของโปรแกรมจัดการอยู่ที่แอดเดรส 3 จะเห็นได้ว่า การเรียกใช้โปรแกรมย่อยนั้น สิ่งที่โปรแกรมหลักต้องรู้ก็คือ มีโปรแกรมย่อยหมายเลขใดบ้าง ในแต่ละหมายเลขมีหน้าที่อะไร ต้องส่งค่าอะไรไปตอนเรียกใช้ และจะได้ค่าอะไรกลับมา เพียงเท่านั้น ไม่จำเป็นต้องรู้รายละเอียดการทำงานเลย

โปรแกรมส่วนควบคุมอินพุท เอ้าท์พุท เป็นโปรแกรมส่วนที่ทำหน้าที่ติดต่อกับฮาร์ดแวร์ ถ้าจะมองในลักษณะที่เป็นโปรแกรมย่อยให้เรียกใช้แล้ว โปรแกรมส่วนนี้ก็จะเป็นเพียงส่วนหนึ่งของโปรแกรมส่วนบริการโปรแกรมย่อย แต่สาเหตุที่แยกโปรแกรมส่วนควบคุมอินพุท เอ้าท์พุท

ออกมาจากโปรแกรมส่วนบริการโปรแกรมย่อย ก็เพื่อ แยกโปรแกรมส่วนที่ทำงานเกี่ยวข้องกับฮาร์ดแวร์ออกมา โปรแกรมส่วนนี้จะขึ้นกับฮาร์ดแวร์ของตู้สาขาโทรศัพท์ ซึ่งจะต้องเขียนใหม่เมื่อมีการนำโปรแกรมไปใช้กับตู้สาขาโทรศัพท์อื่น ดังนั้นการแยกออกมาเป็นส่วนหนึ่งต่างหากก็จะทำให้การแก้ไขโปรแกรมเพื่อนำไปใช้กับตู้สาขาโทรศัพท์อื่น ๆ ต้องทำเฉพาะส่วนควบคุมอินพุท เอ้าท์พุทนี้เท่านั้น โปรแกรมส่วนบริการโปรแกรมย่อยที่เหลือ เป็นอิสระจากฮาร์ดแวร์ ไม่ต้องแก้ไข

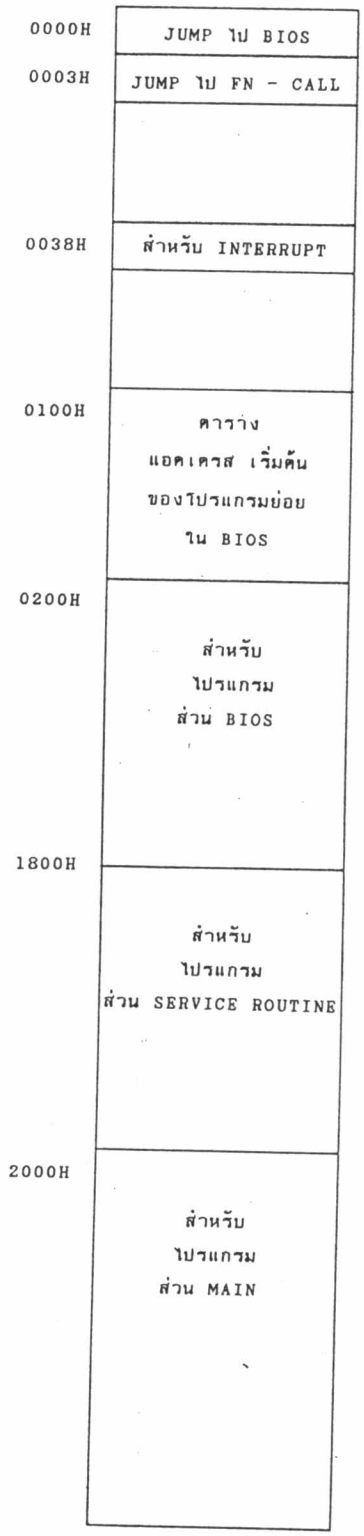


รูปที่ 7.1 การแบ่งโปรแกรมเป็น 3 ส่วน

การแบ่งโปรแกรมเป็น 3 ส่วนนี้ มีตัวอย่างการใช้งานใน โปรแกรมควบคุมของระบบชุมสายโทรศัพท์ ESS#1 และ ในระบบจัดการ CP/M (10) ซึ่งประสบความสำเร็จในการนำโปรแกรมเดียวกันไปใช้ได้ในฮาร์ดแวร์ที่แตกต่างกันมาแล้ว

7.2 การใช้งานหน่วยความจำ

ในตอนก่อนได้แสดงถึงการแยกโปรแกรมที่เกี่ยวข้องกับ การอินพุท เอ้าท์พุท ออกไปไว้ต่างหาก ทำให้โปรแกรมส่วนที่เหลือเป็นอิสระจากอุปกรณ์ อินพุท เอ้าท์พุท และสิ่งที่จะกล่าวถึงต่อไปนี้ คือ แนวทางการทำให้โปรแกรม เป็นอิสระจากหน่วยความจำ ซึ่งจะทำให้โปรแกรมเป็นอิสระจากฮาร์ดแวร์อย่างแท้จริง



รูปที่ 7.2 MEMORY MAP ของการใช้งานหน่วยความจำ ROM

หน่วยความจำมีอยู่ 2 ชนิดคือ ROM และ RAM สำหรับหน่วยความจำ ROM นั้น จะเริ่มต้นที่ แอดเดรส 0 เหมือนกันทุกระบบอยู่แล้วจึงไม่มีปัญหาในการใช้งาน การใช้งานหน่วยความจำ ROM นี้จะมี MEMORY MAP แสดงดังในรูปที่ 7.2

แต่ในกรณีหน่วยความจำ RAM นั้นจะมีปัญหาในการใช้งานอยู่ 2 ข้อ คือ การที่มีจุดเริ่มต้นไม่แน่นอนขึ้นกับฮาร์ดแวร์ของระบบตู้สาขาโทรศัพท์ และอีกปัญหาหนึ่งคือ ความต้องการใช้หน่วยความจำ จะเปลี่ยนไปตามขนาดของตู้สาขาโทรศัพท์ ถ้ามีจำนวนเครื่องรับโทรศัพท์มากก็ต้องการหน่วยความจำมาก ถ้าใช้การกำหนดตำแหน่งต่าง ๆ ในหน่วยความจำอย่างตายตัวจะทำให้โปรแกรมไม่สามารถจะใช้กับตู้สาขาโทรศัพท์ที่มีขนาดต่าง ๆ กันได้ ดังนั้นจึงจำเป็นต้องหาวิธีการในการจัดการหน่วยความจำ

ทางออกทางหนึ่งก็คือ ทำการกันที่ในหน่วยความจำเอาไว้ ให้มีขนาดเพียงพอกับความต้องการใช้หน่วยความจำของกรณีตู้สาขาโทรศัพท์ขนาดใหญ่สุด ไม่ว่าตู้สาขาโทรศัพท์นั้นจะใหญ่หรือเล็ก แต่วิธีนี้จะมีข้อเสียในกรณีที่ตู้สาขาโทรศัพท์มีขนาดเล็ก หน่วยความจำจำนวนมากจะถูกกันที่ไว้แล้ว ไม่ถูกนำไปใช้งาน ทำให้สิ้นเปลือง

แนวทางการแก้ปัญหาในโปรแกรมที่พัฒนาขึ้นก็คือ ทำโปรแกรมสำหรับควบคุมการใช้งานหน่วยความจำขึ้นมาอันหนึ่ง เมื่อต้องการใช้งานหน่วยความจำ ต้องทำการขอจากโปรแกรมควบคุมหน่วยความจำ โปรแกรมอันนี้จะทำการจองที่ในหน่วยความจำเอาไว้เท่ากับจำนวนที่ต้องการ แล้วจะส่งแอดเดรสเริ่มต้นของหน่วยความจำที่ขอไปกลับมาให้ การจัดการหน่วยความจำลักษณะนี้ ได้แนวความคิดมาจาก การจัดสรรหน่วยความจำ แบบต่อเนื่อง (CONTIGUOUS MEMORY ALLOCATION) ในระบบ OS แบบมัลติโปรแกรมมิ่ง (11) การจัดสรรหน่วยความจำโดยวิธีนี้ โปรแกรมหลักไม่จำเป็นต้องรู้จุดเริ่มต้นที่แท้จริงของหน่วยความจำเลย ซึ่งจะทำให้โปรแกรมเป็นอิสระจากหน่วยความจำ

7.3 การแปลงภาษา STL มาเป็นโปรแกรมจริง

เนื่องจากภาษา STL เป็นเพียง PSEUDO CODE ซึ่งใช้ในการอธิบายการทำงานของตู้สาขาโทรศัพท์ แบบสถานะเท่านั้น ยังไม่สามารถนำไปป้อนเป็นคำสั่งให้ไมโครคอมพิวเตอร์ได้ ดังนั้นจึงต้องทำการแปลงโปรแกรมภาษา STL มาเป็นโปรแกรมที่ทำงานได้จริง โดย

จะทำการแปลงมาเป็นภาษา แอสเซมบลี้ ของไมโครโปรเซสเซอร์ เบอร์ Z80

แนวทางในการแปลงภาษา STL ก็คือ ทำการสร้างโปรแกรมย่อยขึ้นมาจำนวนหนึ่ง ให้มีการทำงานตรงกับแต่ละคำสั่งในภาษา STL ซึ่งโปรแกรมย่อยนี้ก็คือ โปรแกรมย่อยในโปรแกรมส่วนบริการโปรแกรมย่อย และส่วนควบคุมอินพุท เอาท์พุท นั่นเอง จากนั้น ทำการแปลงคำสั่งแต่ละบรรทัดของภาษา STL มาเป็นการเรียกใช้โปรแกรมย่อยเหล่านี้ ดังตัวอย่างในรูป 7.3

```
if on_hook ($x) then
  cut    ($icm,$x)
  tone   ($icm,stop)
  close  ($icm)
  goto IDEL {3}
end
```

```
EXAMPLE:
        LD      A,38                    ; on_hook
        LD      E,(IX+X)                ; $x
        CALL    3

        DEC     L                       ; if
        JP      Z,NEXT

        LD      A,23                    ; cut
        LD      D,(IY+$ICM)             ; $icm
        LD      E,(IX+X)                ; $x
        CALL    3

        LD      A,37                    ; tone
        LD      D,(IY+$ICM)             ; $icm
        LD      E,0                     ; stop
        CALL    3

        LD      A,129                   ; close
        LD      D,(IY+$ICM)             ; $icm
        CALL    3

        LD      A,32                    ; เริ่มจับเวลาใหม่
        LD      E,(IX+X)
        CALL    3

        LD      (IY+$STATE  ),
        LD      (IY+$STATE+1), แอดเดรสเริ่มต้นของโปรแกรมประจำสถานะ IDLE

NEXT:
```

รูปที่ 7.3 การแปลงภาษา STL

จากรูปที่ 7.3 จะเห็นได้ว่าจากภาษา STL หนึ่งคำสั่ง (หนึ่งบรรทัด) จะถูกแปลงมาเป็นการเรียกใช้โปรแกรมย่อยหนึ่งครั้ง โดยที่การเรียกใช้โปรแกรมย่อยในแต่ละครั้งนั้นจะประกอบด้วย การกำหนดหมายเลขของโปรแกรมย่อยที่ต้องการ การส่งค่าพารามิเตอร์ไปยังโปรแกรมย่อยโดยผ่านทางรีจิสเตอร์ แล้วจึง เรียกใช้โปรแกรมย่อยโดยผ่านทางโปรแกรมจัดการที่อยู่ในโปรแกรมส่วนบริการโปรแกรมย่อย ทำให้ภาษา STL หนึ่งบรรทัดถูกแปลงมาเป็นภาษาแอสเซมบลี้หลายบรรทัด

เพื่อให้สามารถแปลงภาษา STL มาเป็นภาษาแอสเซมบลี้ เป็นไปในลักษณะบรรทัดต่อ บรรทัด จึงได้นำ MACRO มาใช้ โดยทำการแทน การกำหนดหมายเลขโปรแกรม การส่งค่าพารามิเตอร์ การเรียกไปยังโปรแกรมจัดการ ด้วย MACRO บรรทัดเดียว ดังตัวอย่างในรูปที่ 7.4

```
X_JLCOFF:       MACRO   #LC,#LABEL
                LD      A,38
                LD      E,#LC
                CALL    3

                DEC     L
                JP      NZ,LABEL
                EMDM

X_CUT:          MACRO   #ICM,#LC
                LD      A,23
                LD      D,#ICM
                LD      E,#LC
                CALL    3
                ENDM

X_TONE:         MACRO   #ICM,#TONE
                LD      A,37
                LD      D,#ICM
                LD      E,#TONE
                CALL    3
                ENDM

X_CLOSE:        MACRO   #ICM
                LD      A,39
                LD      E,(IY+$ICM)
                CALL    3
                ENDM

X_GOTO:         MACRO   #LC,#STATE
                LD      A,32
                LD      E,#LC
                CALL    3

                LD      (IY+$STATE  ),LO #STATE
                LD      (IY+$STATE+1),HI #STATE
                ENDM
```

รูปที่ 7.4 การใช้ MACRO ในการแปลงภาษา STL

```
EXAMPLE:
        X_JLCOFF        (IX+X),NEXT             ; if on_hook ($x) then
        X_CUT           (IY+$ICM),(IX+X)        ;   cut    ($icm,$x)
        X_TONE          (IY+$ICM),0             ;   tone   ($icm,stop)
        X_CLOSE         (IY+$ICM)               ;   close  ($icm)
        X_GOTO          (IX+X),IDLE             ;   goto IDEL (3)
NEXT:
```

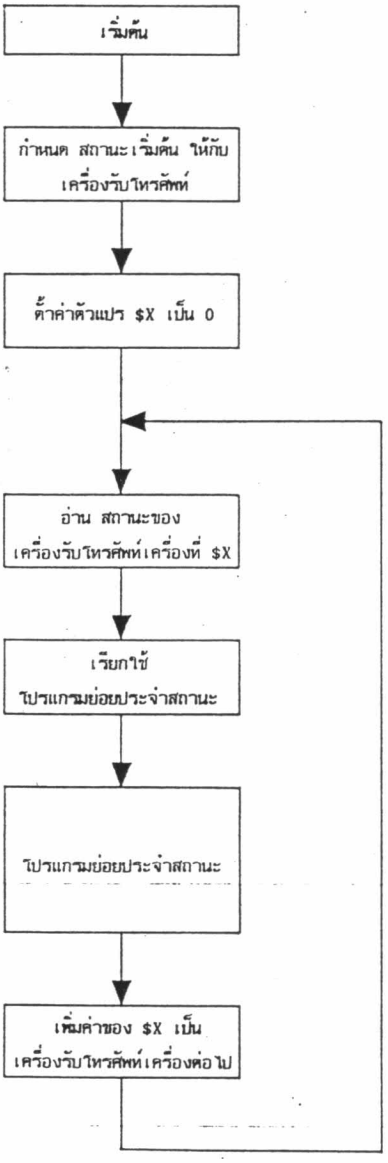
รูปที่ 7.4 การใช้ MACRO ในการแปลงภาษา STL (ต่อ)

7.4 การแบ่งเวลาให้เครื่องรับโทรศัพท์แต่ละเครื่อง

เท่าที่ผ่านมา การอธิบายการทำงานของตู้สาขาโทรศัพท์โดยไดอะแกรมสถานะ และภาษา STL เป็นการอธิบายการทำงานโดยยึดเครื่องรับโทรศัพท์เพียงเครื่องเดียว แต่ในความเป็นจริงนั้น ตู้สาขาโทรศัพท์ต้องคอยควบคุมการทำงานของเครื่องรับโทรศัพท์ทุกเครื่อง นั่นคือตู้สาขาโทรศัพท์มีการทำงานแบบมัลติโปรแกรมมิ่ง คือต้องแบ่งเวลาการทำงานให้กับเครื่องรับโทรศัพท์แต่ละเครื่อง

วิธีการแบ่งเวลาการทำงานในโปรแกรมที่พัฒนาขึ้นทำโดย ในส่วนของโปรแกรมหลักจะมีส่วนที่ควบคุมการทำงานแบบสถานะอยู่ การทำงานของโปรแกรมควบคุมนี้ จะทำงานร่วมกับตัวแปร $X (ใช้ชื่อตามภาษา STL) ซึ่งจะบอกโปรแกรมประจำสถานะให้รู้ว่า ขณะนี้ ตู้สาขาโทรศัพท์ทำงานกับเครื่องรับโทรศัพท์เครื่องใดอยู่ ขั้นตอนการทำงานของโปรแกรมควบคุม เป็นดังรูป 7.5 คือ ในขั้นแรก ทำการกำหนดค่า $X เป็นเครื่องรับโทรศัพท์เครื่องแรก (เครื่องที่ 0) แล้วทำการอ่านสถานะของโทรศัพท์เครื่องที่ $X มาดูว่าอยู่ในสถานะใด แล้วทำการเรียกโปรแกรมประจำสถานะมาทำงาน จากค่า $X เป็นเครื่องที่ 0 โปรแกรมประจำสถานะก็จะทำ

งานกับโทรศัพท์เครื่องที่ 0 แล้วทำการเพิ่มค่า $X เป็นเครื่องที่ 1 2 3 เรื่อยไป เพื่อทำงานกับเครื่องรับโทรศัพท์เหล่านั้น จนครบทุกเครื่องก็จะวนไปทำเครื่องที่ 0 อีก



รูปที่ 7.5 การทำงานของโปรแกรมควบคุม