บทที่ 2

การจัดการหน่วยความจำและโปรแกรมภายใต้ระบบเอมเอสดอส

การจัดโครงสร้างหน่วยความจำของระบบเอมเอสดอส

เมื่อเครื่องคอมพิวเตอร์เริ่มทำงาน โปรแกรมจะเริ่มทำงานที่ตำแหน่ง ØFFFFØh ของหน่วยความจำ ซึ่งเป็นลักษณะที่ออกแบบมาสำหรับไมโครโปรเซสเซอร์ตะกูล 8086 ตำแหน่งหน่วยความจำดังกล่าว ได้แก่ ส่วนที่เรียกว่า รอม (ROM มาจาก Read Only Memory) ภายในบริเวณนี้ ประกอบด้วย การทำงานที่เกี่ยวกับการทดสอบระบบ และ การทำงานของโปรแกรมบูตแสตรปของรอม (ROM Bootstrap)

โปรแกรมบูตแสตรปของรอม จะทำการอ่านระเบียนข้อมูลที่อยู่ในตำแหน่งเซกเตอร์แรก หรือเรียกว่า Boot Sector ของแผ่นจานบันทึก ซึ่งก็คือ โปรแกรมบูตแสตรปเช่นกัน โปรแกรมบูตแสตรปของจานบันทึก จะถูกบรรจุลงในหน่วยความจำ และทำการควบคุมการทำงานของระบบต่อไป

โปรแกรมบูตแสตรปของจานบันทึก จะตรวจสอบเพื่อค้นหาแฟ้มข้อมูลที่มีในเซกเตอร์แรกของไดเรคทอรีราก (Root Directory) แฟ้มทั้งสองได้แก่ IO.SYS และ MSDOS.SYS เรียงตามลำดับ (สำหรับระบบพีซีดอส (PC-DOS) ได้แก่ IBMIO.COM และ IBMDOS.COM) แฟ้มทั้งสองจะถูกอ่านและบรรจุในหน่วยความจำ จากนั้นการทำงานจะเริ่มที่ตำแหน่งเริ่มต้นการทำงานของชุดคำสั่งภายใน IO.SYS

ชุดคำสั่งภายใน IO.SYS ประกอบด้วยโมดูลการทำงาน 2 ส่วน ส่วนแรก เรียกว่า ไบออส (BIOS มาจาก BASIC Input/Output System) ประกอบด้วย ชุดการทำงานที่เชื่อมต่อกันของดิไวส์ไดร์ฟเวอร์ (Device Driver) ที่ประจำอยู่ในระบบ ได้แก่ คอนโซล (Console) พอร์ตเสริม (Auxillary Port) เครื่องพิมพ์ (Printer) เป็นต้น และการทำงานที่กำหนดสภาพเริ่มต้นของอุปกรณ์บางชิ้น ซึ่งจะทำงานเฉพาะตอนเริ่มต้นการทำงานของระบบเท่านั้น ส่วนที่สองของ IO.SYS เรียกว่า SYSINIT

SYSINIT จะทำการตรวจสอบจำนวนหน่วยความจำทั้งหมดที่ต่อเนื่องกันของระบบในขณะนั้น และโยกย้ายชุดคำสั่งที่เป็นของ SYSINIT จากหน่วยความจำที่ตำแหน่งเดิมไปบรรจุในตำแหน่งหน่วยความจำที่สูง(high memory address) จากนั้นจึงย้ายส่วนที่เรียกว่า DOS

Kernel ของ MSDOS.SYS จากตำแหน่งเดิมบรรจุทับในตำแหน่งที่เคยเป็นของ SYSINIT

ต่อจากนั้น SYSINIT จะเรียกใช้การทำงานจาก DOS Kernel เพื่อกำหนดสภาพเริ่มต้นของระบบ การทำงานของ DOS Kernel ประกอบด้วย การกำหนดตารางภายใน และเนื้อที่หน่วยความจำสำหรับการทำงานของระบบ การกำหนดตำแหน่งเวคเตอร์สำหรับอินเทอร์รัปต์หมายเลข 20h ถึง 2Fh และการกำหนดสภาพเริ่มต้นสำหรับดิไวส์ไดร์ฟเวอร์ที่ประจำในระบบ ซึ่งมีการตรวจสอบสภาพของอุปกรณ์ และการกำหนดสภาพเริ่มต้นสำหรับอุปกรณ์ภายนอกที่จำเป็นบางชิ้น รวมถึงกำหนดตำแหน่งเวคเตอร์สำหรับการอินเทอร์รัปต์จากอุปกรณ์นั้น

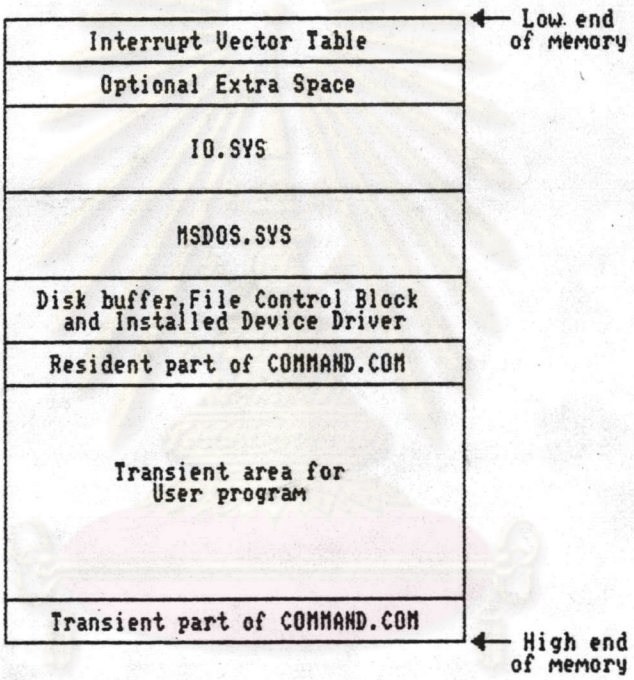
เมื่อ DOS Kernel กำหนดสภาพเริ่มต้นภายในระบบเรียบร้อย และดิไวส์ไดร์ฟเวอร์มีสภาพพร้อมสำหรับการทำงาน SYSINIT จะทำการเปิดแฟ้มข้อมูลที่มีชื่อว่า CONFIG.SYS ที่มีในจานบันทึก ซึ่งเป็นแฟ้มข้อมูลที่ผู้ใช้สามารถกำหนดขึ้นเองได้ โดยประกอบด้วยคำสั่งที่ให้ความสะดวกต่อผู้ใช้ในการกำหนดสภาพแวดล้อมของระบบเอมเอสดอส (MSDOS environment) ตัวอย่างเช่น การติดตั้งดิไวส์ไดร์ฟเวอร์จากภายนอกระบบ การกำหนดจำนวนบัฟเฟอร์สำหรับข้อมูลที่อ่านมาจากจานบันทึก และการกำหนดจำนวนแฟ้มข้อมูลที่มากที่สุดซึ่งสามารถเปิดใช้ได้ในเวลาเดียวกัน เป็นต้น หลังจากการติดตั้งระบบจากคำสั่งภายในแฟ้ม CONFIG.SYS เรียบร้อย SYSINIT จะทำการปิดแฟ้มข้อมูลที่เกี่ยวข้องทั้งหมดและเปิดใช้เฉพาะ คอนโซล (CON) เครื่องพิมพ์(PRN) อุปกรณ์เสริม(AUX) และแฟ้มมาตราฐาน ได้แก่ standard input standard output และ standard error

ขั้นตอนสุดท้าย SYSINIT จะทำการบรรจุและปฏิบัติงานโปรแกรมประมวลผลคำสั่ง (Command Processor) ซึ่งได้แก่ COMMAND.COM หลังจากนั้นการทำงานของระบบจะถูกส่งมาที่โปรแกรมประมวลผลคำสั่ง ซึ่งจะเป็นโปรแกรมที่นำโปรแกรมต่าง ๆ มาปฏิบัติงานในคราวต่อไป เมื่อนำโปรแกรมประมวลผลคำสั่งบรรจุในหน่วยความจำ โปรแกรมจะแบ่งออกเป็น 3 ส่วน คือ ส่วนที่ประจำในระบบ(resident portion) ส่วนกำหนดการเริ่มต้น(initialization section) และโมดูลชั่วคราว(transient module)

ส่วนที่ประจำในระบบ เป็นโมดูลที่บรรจุในตำแหน่งที่ถัดไปจาก DOS Kernel ประกอบด้วย โมดูลการทำงานที่จะเกิดขึ้นหลังจากการกดปุ่ม Ctrl-C หรือ Ctrl-Break โมดูลการทำงานเมื่อเกิดความผิดพลาดวิกฤต(Critical error) และโมดูลการทำงานหลังจากจบการทำงานของโปรแกรมต่าง ๆ

ส่วนกำหนดการเริ่มต้น เป็นโมดูลที่จะถูกบรรจุในตำแหน่งที่ถัดจาก ส่วนที่ประจำในระบบในตอนเริ่มต้นของระบบ เพื่อทำหน้าที่ประมวลผลคำสั่งต่าง ๆ ที่มีอยู่ในแฟ้ม AUTOEXEC.BAT และจะถูกยกเลิกหลังจากจบการทำงานของคำสั่งที่มีในแฟ้ม

โมดูลชั่วคราว เป็นโมดูลที่ถูกบรรจุในตำแหน่งหน่วยความจำที่สูง ทำหน้าที่รับคำสั่งที่ป้อนทางคีย์บอร์ด และนำมาปฏิบัติงาน หน่วยความจำบริเวณนี้อาจถูกบรรจุทับโดยโปรแกรมที่นำมาปฏิบัติงาน ส่วนที่ประจำในระบบ จึงต้องทำหน้าที่ตรวจสอบการถูกบรรจุทับจากโปรแกรมอื่น เพื่อบรรจุโมดูลส่วนนี้กลับคืนในหน่วยความจำหลังจากจบการทำงานของโปรแกรมนั้น



ภาพที่ 2.1 โครงสร้างหน่วยความจำ หลังจากระบบเอมเอสดอสพร้อมปฏิบัติการ

<u>ฟังก์ชันจัดการหน่วยความจำของระบบเอมเอสดอส</u>

หลังจากโปรแกรมประมวลผลคำสั่งเข้าควบคุมการทำงานทั้งหมด และรอการป้อนคำสั่งหรือโปรแกรมปฏิบัติงานจากผู้ใช้ ขณะนั้นเนื้อที่หน่วยความจำได้แบ่งออกเป็น 2 บริเวณ คือ เนื้อที่ของระบบเอมเอสดอส และ เนื้อที่ชั่วคราวสำหรับโปรแกรม (transient program area) ที่ใช้ปฏิบัติงาน ซึ่งสามารถเปลี่ยนแปลงขนาดของหน่วยความจำที่ได้จาก

การทำงานของโปรแกรม สิ่งที่โปรแกรมประมวลผลคำสั่งต้องทำ เมื่อกำหนดให้โปรแกรมเข้าปฏิบัติงาน คือ การกำหนดเนื้อที่ว่างแห่งแรกให้กับบล็อกข้อมูลพิเศษสำหรับโปรแกรม เรียกว่า พีเอสพี (PSP มาจาก Program Segment Prefix)ก่อนการกำหนดเนื้อที่ที่เหลือทั้งหมดให้กับโปรแกรม พีเอสพี เป็นบล็อกข้อมูลที่มีขนาด 256 ไบต์ ประกอบด้วย ตำแหน่งที่เก็บสารสนเทศที่มีความจำเป็นสำหรับโปรแกรมในระหว่างที่ปฏิบัติงาน หลังจากนั้นจึงส่งการทำงานไปที่จุดเริ่มต้นการทำงานของโปรแกรม เมื่อโปรแกรมจำเป็นต้องกำหนดเนื้อที่หน่วยความจำเพิ่มเติม เพื่อใช้เป็นบัฟเฟอร์ชั่วคราวในระหว่างการทำงาน โปรแกรมจะต้องเตรียมเนื้อที่ให้ว่างพอสำหรับขนาดหน่วยความจำที่ต้องการ โดยการลดขนาดเนื้อที่ให้มีขนาดเท่าที่จำเป็นสำหรับโปรแกรมนั้น เนื้อที่ว่างที่เกิด จึงสามารถกำหนดจองได้จากโปรแกรม เมื่อจบการทำงานของโปรแกรม เนื้อที่เหล่านั้นและเนื้อที่ของโปรแกรมจะถูกยกเลิกและคืนให้กับระบบ เพื่อใช้กับโปรแกรมอื่นต่อไป การทำงานที่เกี่ยวกับการจัดเนื้อที่หน่วยความจำ จึงแบ่งออกเป็น 3 ลักษณะ คือ

1. การจองเนื้อที่หน่วยความจำ (Allocating memory area)
2. การปลดปล่อยเนื้อที่ซึ่งถูกจอง (Releasing memory area)
3. การเปลี่ยนแปลงขนาดเนื้อที่หน่วยความจำ (Modifying memory area)

โปรแกรมที่ต้องการใช้การทำงานทั้ง 3 แบบนี้ สามารถเรียกใช้ได้จากฟังก์ชันระบบของเอมเอสดอส โดยการใช้คำสั่งอินเทอรรัปต์ หมายเลข 21h ซึ่งระบุหมายเลขฟังก์ชันลงในรีจิสเตอร์ AH สำหรับฟังก์ชันที่ใช้ ได้แก่ ฟังก์ชันหมาย เลข 48h ทำหน้าที่ในการจองเนื้อที่หน่วยความจำ ฟังก์ชันหมายเลข 49h ทำหน้าที่ในการปล่อยเนื้อที่หน่วยความจำ และฟังก์ชันหมายเลข 4Ah ทำหน้าที่ในการเปลี่ยนแปลงขนาดเนื้อที่หน่วยความจำที่จอง เมื่อเกิดการผิดพลาดในการทำงาน จะกำหนดค่าให้กับแคร์รี่แฟลก(carry flag) และ กำหนดรหัสความผิดพลาดไว้ภายในรีจิสเตอร์ AX

ระบบเอมเอสดอส สามารถควบคุมการจองเนื้อที่หน่วยความจำให้กับโปรแกรมต่าง ๆ ได้ โดยการสร้างบล็อกควบคุมหน่วยความจำ(memory control block) เพื่อใช้ควบคุมเนื้อที่ต่าง ๆ ที่ถูกกำหนดขึ้นในหน่วยความจำ บล็อกควบคุมหน่วยความจำเป็นเนื้อที่หน่วยความจำส่วนหนึ่งที่เอมเอสดอสสร้างขึ้น ก่อนที่กำหนดเนื้อที่ให้กับโปรแกรม แต่ละกลุ่มของหน่วยความจำที่กำหนดขึ้น จึงถูกคั่นด้วยบล็อกควบคุมหน่วยความจำ มีขนาด 16 ไบต์ ภายใน

บล๊อกควบคุมหน่วยความจำ ประกอบด้วย

1. รหัสควบคุม เป็นแฟลกที่ใช้ตรวจสอบ เพื่อระบุว่า เนื้อที่หน่วยความจำภายใต้บล๊อกนี้ถูกกำหนดจากระบบ ขนาดของรหัสควบคุมเท่ากับ 1 ไบต์ ค่าของรหัสที่ใช้ได้แก่ 4Dh หมายความว่า เนื้อที่บริเวณนี้เป็นส่วนหนึ่งในการเชื่อมต่อของเนื้อที่ทั้งหมดที่ถูกกำหนดจากระบบ และ 5Ah หมายความว่า เป็นเนื้อที่บล๊อกสุดท้ายของการเชื่อมต่อ

2. เขตข้อมูลที่ระบุว่า เป็นเนื้อที่หน่วยความจำที่ว่าง หรือเป็นเนื้อที่ซึ่งถูกจองจากโปรแกรม โดยเก็บตำแหน่งเซกเมนต์ (segment) ของพีเอสพีของโปรแกรม เขตข้อมูลนี้มีขนาด 2 ไบต์

3. เขตข้อมูลที่บันทึกขนาดเนื้อที่ซึ่งอยู่ในการควบคุมของบล๊อก ขนาดเนื้อที่หน่วยความจำที่บันทึก มีหน่วยเป็นพารากราฟ (paragraph) โดยที่ 1 พารากราฟ มีขนาดเท่ากับ 16 ไบต์ เขตข้อมูลนี้มีขนาด 2 ไบต์

เมื่อโปรแกรมต้องการกำหนดเนื้อที่หน่วยความจำเพิ่มเติม โดยการใช้ฟังก์ชันจองเนื้อที่หน่วยความจำของระบบเอมเอสดอส ฟังก์ชันจะทำการค้นหาบล๊อกควบคุมหน่วยความจำซึ่งมีขนาดมากเพียงพอสำหรับขนาดที่โปรแกรมต้องการ แล้วจึงทำการบันทึกตำแหน่งของพีเอสพีของโปรแกรมในเขตข้อมูลที่แสดงการจับจอง และขนาดของเนื้อที่ที่ต้องการในบล๊อกควบคุมหน่วยความจำ ถ้าเนื้อที่หน่วยความจำนั้นมีขนาดมากกว่าขนาดที่ต้องการ ฟังก์ชันระบบจะสร้างบล๊อกควบคุมหน่วยความจำใหม่ สำหรับหน่วยความจำที่เหลือจากการจองและกำหนดรหัสควบคุม เพื่อให้อยู่ในลักษณะการเชื่อมต่อระหว่างเนื้อที่หน่วยความจำสำหรับใช้ในจัดสรรเนื้อที่คราวต่อไป เมื่อฟังก์ชันกำหนดเนื้อที่หน่วยความจำเรียบร้อย จะส่งตำแหน่งเซกเมนต์ของหน่วยความจำที่จอง กลับมาให้กับโปรแกรม โดยระบุไว้ในรีจิสเตอร์ AX ลักษณะการใช้ฟังก์ชัน 48h เป็นดังนี้

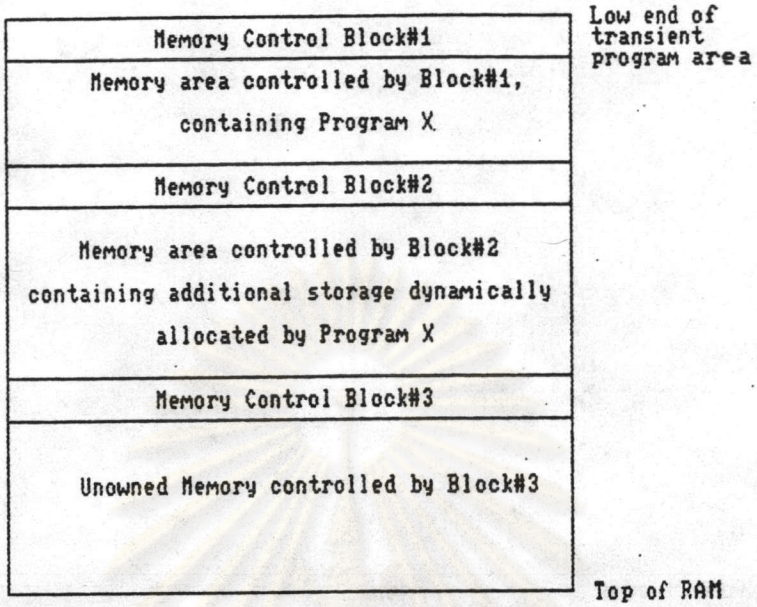
```
 :          :
mov         BX,size   ; the requested size is in paragraph
mov         AH,48h    ; allocate memory
int         21h       ; if failed, BX is the largest block
 :            :       ; available
```

เมื่อโปรแกรมไม่ต้องการใช้เนื้อที่หน่วยความจำนั้นอีกต่อไป และต้องการปล่อยเนื้อที่ออกจากการจอง ฟังก์ชันระบบจะทำการปล่อยเนื้อที่ส่วนนั้น โดยการแก้ไขเซตข้อมูลที่แสดงถึงการจับจองหน่วยความจำ ซึ่งบรรจุตำแหน่งเซกเมนต์ของพีเอสพีของโปรแกรมภายในบล๊อกควบคุมหน่วยความจำ ให้มีค่าเท่ากับ Ø หน่วยความจำของบล๊อกควบคุมหน่วยความจำนั้น จะเป็นหน่วยความจำที่ว่างสำหรับการจับจองในครั้งต่อไป ลักษณะการใช้ฟังก์ชัน 49h เป็นดังนี้

```
 :          :
mov         ES,block ; ES is segment address to be freed
mov         AH,49h   ; Free allocated memory block
int         21h
 :          :
```

แต่เมื่อโปรแกรมต้องการเปลี่ยนแปลงขนาดเนื้อที่หน่วยความจำ ซึ่งมีอยู่ในขณะนั้นโดยการลดขนาดหน่วยความจำ ฟังก์ชันระบบจะสร้างบล๊อกควบคุมใหม่สำหรับเนื้อที่หน่วยความจำที่เหลือจากการลดขนาดเช่นเดียวกับการจองเนื้อที่หน่วยความจำ และเพิ่มการเชื่อมต่อของบล๊อกหน่วยความจำที่เกิดขึ้นใหม่ แต่ถ้าโปรแกรมต้องการขยายเนื้อที่หน่วยความจำ ฟังก์ชันระบบจะทำการรวบรวมเนื้อที่ว่าง ซึ่งอยู่ติดต่อกับเนื้อที่หน่วยความจำนั้นและแก้ไขขนาดเนื้อที่ให้มีขนาดเพิ่มขึ้นตามที่ต้องการ จากนั้นจึงส่งตำแหน่งเซกเมนต์ของหน่วยความจำที่เปลี่ยนแปลงขนาดกลับมาให้โปรแกรม ลักษณะการใช้ฟังก์ชัน 4Ah เป็นดังนี้

```
 :          :
mov         ES,block    ; ES is segment address to be modified
mov         BX,newsize  ; BX is the requested size
mov         AH,4Ah      ; modify the allocated memory block
int         21h
 :          :
```



ภาพที่ 2.2 การควบคุมเนื้อที่หน่วยความจำจากบล๊อกควบคุมหน่วยความจำ

<u>การบรรจุโปรแกรมประจำในหน่วยความจำ</u>

ตามปกติระบบเอมเอสดอส จะกำหนดเนื้อหน่วยความจำชั่วคราวทั้งหมดให้กับโปรแกรมที่นำมาปฏิบัติการ และปล่อยเนื้อที่เหล่านั้นคืนให้กับระบบเมื่อจบการทำงานของโปรแกรม เพื่อสามารถกำหนดให้กับโปรแกรมอื่นได้ใช้ปฏิบัติงานในครั้งต่อไป แต่เมื่อต้องการให้คำสั่งการทำงานเหล่านั้นยังคงอยู่ในหน่วยความจำ โปรแกรมการทำงานนั้นจะต้องลดเนื้อที่หน่วยความจำให้เหลือมากพอสำหรับโปรแกรมอื่นจะใช้ปฏิบัติงานได้ และจบการทำงานของโปรแกรมนั้น เอมเอสดอสได้เตรียมวิธีจบการทำงานให้กับผู้เขียนโปรแกรมโดยยังคงเก็บฟังก์ชันของโปรแกรมไว้ในหน่วยความจำ วิธีจบการทำงานดังกล่าวมีอยู่2 วิธีคือ

1. การจบการทำงานของโปรแกรม โดยการใช้คำสั่งอินเทอร์รัปต์หมายเลข 27h เป็นการจบการทำงานที่สงวนเนื้อที่หน่วยความจำตั้งแต่ ตำแหน่งเริ่มต้นของโปรแกรมจนถึง ตำแหน่งสุดท้ายที่เซกเมนต์และออฟเซต ซึ่งระบุด้วยรีจิสเตอร์ CS:DX ไม่ให้ถูกใช้จาก

โปรแกรมอื่นที่นำมาปฏิบัติงานในภายหลัง เนื้อที่ที่สงวนไว้มีขนาดไม่เกิน 64 กิโลไบต์ ลักษณะของโปรแกรมเป็นดังนี้

```
Program   SEGMENT
            :          :
          mov        DX,offset Last_byte  ; get the last byte
          int        27h                  ; terminate & stay resident
            :          :
Last_byte:
Program   ENDS
          END        Start
```

2. การจบการทำงานของโปรแกรม โดยการเรียกใช้ฟังก์ชันระบบของเอมเอสดอส ซึ่งเป็นฟังก์ชันหมายเลข 31h ทำหน้าที่จบการทำงานของโปรแกรม และสงวนเนื้อที่ของโปรแกรมในหน่วยความจำ กำหนดขนาดเท่ากับจำนวนที่ระบุในรีจิสเตอร์ DX (มีหน่วยเป็นพารากราฟ) โดยเริ่มจากจุดเริ่มต้นของโปรแกรมที่ถูกบรรจุในหน่วยความจำ ข้อดีสำหรับวิธีการนี้คือ สามารถเก็บเนื้อที่ไว้ได้มากกว่า 64 กิโลไบต์ และจะไม่ยกเลิกเนื้อที่หน่วยความจำ ซึ่งถูกจองจากโปรแกรมนั้น ลักษณะของโปรแกรมเป็นดังนี้

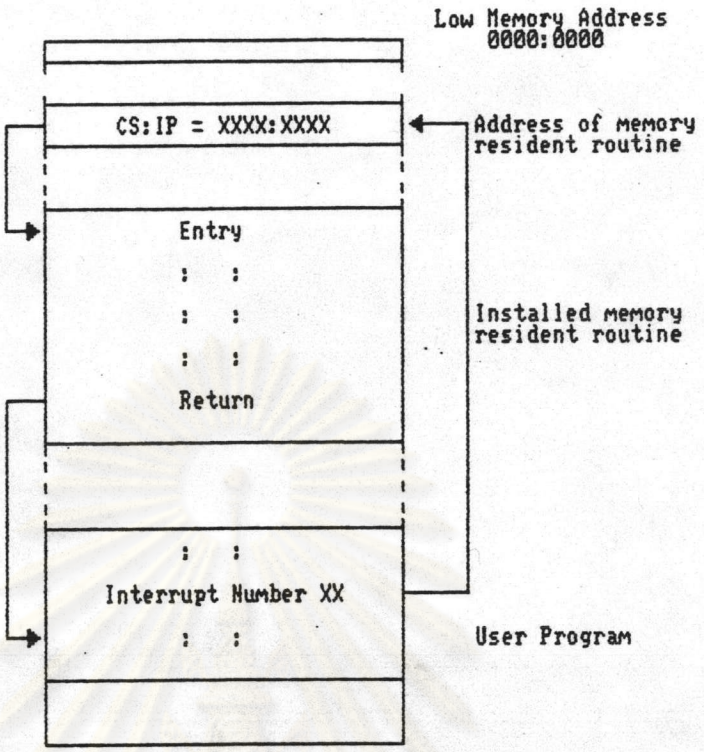
```
Program   SEGMENT
            :          :
          mov        AX,PSP               ; get PSP segment address
          mov        DX,seg End_Addr      ; get the last segment address
          sub        DX,AX                ; differnce is the program size
          mov        ah,31h               ; Keep process in memory
          int        21h
            :          :
Program   ENDS
```

```
End_Addr   SEGMENT
End_Addr   ENDS
           END        Start
```

<u>การเรียกใช้ฟังก์ชันของโปรแกรมที่ประจำในหน่วยความจำ</u>

เมื่อโปรแกรมถูกติดตั้งในหน่วยความจำ สิ่งที่ต้องคำนึงถึงต่อไปคือ การติดต่อเพื่อเรียกใช้ฟังก์ชันที่มีในหน่วยความจำ สำหรับระบบเอมเอสดอสไม่ได้กำหนดวิธีการใด ๆ ให้กับผู้เขียนโปรแกรมใช้ติดต่อการทำงานเหล่านั้นหลังจากที่จบการทำงานของโปรแกรม ในการเตรียมหนทางสำหรับติดต่อกับฟังก์ชันของโปรแกรมที่อยู่ในหน่วยความจำ จะต้องรู้ตำแหน่งของหน่วยความจำที่เป็นจุดเริ่มต้นการทำงานของฟังก์ชันเหล่านั้น และต้องมีเนื้อที่หน่วยความจำส่วนหนึ่งซึ่งโปรแกรมทั่วไปสามารถเข้าถึงได้ และไม่ส่งผลกับการทำงานของระบบ เพื่อใช้เป็นเนื้อที่สำหรับเก็บตำแหน่งเริ่มต้นของการทำงาน เนื้อที่หน่วยความจำที่โปรแกรมสามารถเข้าถึงได้สะดวก คือ เนื้อที่ของตารางอินเทอร์รัปต์เวคเตอร์ (Interrupt Vector Table) เป็นตารางที่มีขนาด 1024 ไบต์ ใช้อ้างอิงถึงการทำงานต่าง ๆ ที่มีอยู่ในระบบจากตำแหน่งเซกเมนต์และออฟเซต ที่มีขนาด 4 ไบต์ จึงทำให้ภายในตารางสามารถเก็บหมายเลขการอินเทอร์รัปต์ได้ถึง 256 หมายเลข โดยที่ 64 หมายเลขแรก ได้แก่ 00h ถึง 3Fh ใช้สำหรับการอินเทอร์รัปต์ที่เกี่ยวข้องกับการทำงานของฮาร์ดแวร์ และการอินเทอร์รัปต์เพื่อเรียกใช้ฟังก์ชันระบบของเอมเอสดอส สำหรับหมายเลขที่เหลือ 192 หมายเลข ได้แก่ 40h ถึง FFh เป็นบริเวณที่ผู้เขียนโปรแกรม สามารถใช้เป็นเนื้อที่สำหรับเก็บตำแหน่งของฟังก์ชันที่อยู่ในหน่วยความจำได้ ดังนั้นวิธีการเรียกใช้ฟังก์ชันของโปรแกรมที่ประจำในหน่วยความจำจึงสามารถทำได้โดยการอินเทอร์รัปต์ด้วยหมายเลขอินเทอร์รัปต์เวคเตอร์ ซึ่งใช้เก็บตำแหน่งเริ่มต้นของฟังก์ชันในโปรแกรม

แต่ในระหว่างการใช้งาน ผู้ใช้ไม่สามารถทราบได้ว่ามีการติดตั้งโปรแกรมให้ประจำในหน่วยความจำขณะนั้นหรือไม่ ซึ่งอาจส่งผลเสียต่อการทำงานของระบบทั้งหมดได้ ถ้ามีการเรียกใช้ โดยไม่มีการติดตั้งการทำงานในหน่วยความจำ ดังนั้นก่อนการเรียกใช้ฟังก์ชันที่ประจำในหน่วยความจำ จึงจำเป็นต้องมีการตรวจสอบการติดตั้งโปรแกรมให้ประจำในหน่วยความจำ วิธีการตรวจสอบการติดตั้ง สามารถทำได้ 2 วิธีคือ



ภาพที่ 2.3 ลักษณะการเรียกใช้ฟังก์ชันจากโปรแกรมที่ประจำในหน่วยความจำ

1. อ่านรหัสควบคุม (control code) หรือข้อมูลซึ่งอยู่ในลักษณะข้อความที่รับรู้ได้หลังจากการเปรียบเทียบค่า รหัสควบคุม หรือข้อความเหล่านี้จะต้องอยู่ในบริเวณใกล้เคียง หรือมีระยะห่างที่คงที่กับจุดเริ่มต้นการทำงาน หลังจากที่ติดตั้งโปรแกรมในหน่วยความจำ โดยที่โปรแกรมทั่วไป สามารถอ่านตำแหน่งเริ่มต้นการทำงานจากตารางอินเทอร์รัปต์เวคเตอร์ได้ โดยการใช้ฟังก์ชันระบบของเอมเอสดอส หมายเลขฟังก์ชัน 35h

2. โดยการตรวจสอบข้อมูลที่มีในตารางอินเทอร์รัปต์เวคเตอร์ จากหมายเลขที่ไม่ได้กำหนดใช้ภายในระบบ (ได้แก่ 40h ถึง FFh) โดยทั่วไป ค่าของข้อมูลภายในช่วงหมายเลขอินเทอร์รัปต์เวคเตอร์เหล่านี้ จะเป็นค่าที่ตรวจสอบได้คือ มีค่าเท่ากันทั้งหมด ซึ่งอาจมีค่าเท่ากับ 00h เหมือนกันทั้งหมด หรือมีค่าเท่ากับ FFh เหมือนกันทั้งหมด ทำให้สามารถแยกการติดตั้งโปรแกรมได้จากความแตกต่างที่พบ แต่นับว่าเป็นวิธีที่ค่อนข้างไม่แน่นอนเหมือนกับวิธีแรก

การยกเลิกโปรแกรมที่ประจำในหน่วยความจำ

เมื่อฟังก์ชันของโปรแกรมที่ประจำในหน่วยความจำ ไม่จำเป็นต่อการเรียกใช้ของโปรแกรมปฏิบัติงานอีกต่อไป จึงควรมีวิธีการยกเลิกฟังก์ชันเหล่านั้น และนำออกจากหน่วยความจำ เพื่อให้โปรแกรมมีเนื้อที่หน่วยความจำเพิ่มขึ้นสำหรับใช้งานในคราวต่อไป วิธีการยกเลิกการทำงานที่ง่ายที่สุดก็คือ การเริ่มต้นระบบใหม่(reboot system) ซึ่งจะเป็นการลบตำแหน่งที่ถูกติดตั้งในครั้งแรกออกจากตารางอินเทอร์รัปต์เวคเตอร์ และเป็นการจัดเนื้อที่หน่วยความจำของระบบขึ้นใหม่ แต่เป็นวิธีที่ไม่สะดวกและไม่ควรใช้บ่อยนัก สำหรับวิธีการที่สามารถยกเลิกการทำงานเหล่านั้นได้อย่างสมบูรณ์ จากการปฏิบัติงานของโปรแกรมโดยตรง โปรแกรมปฏิบัติงานนั้น จะต้องประกอบด้วยขั้นตอนการทำงานดังนี้

1. ยกเลิกการติดต่อของฟังก์ชันกับโปรแกรมภายนอก ด้วยการลบตำแหน่งที่ติดตั้งนั้นออกจากตารางอินเทอร์รัปต์เวคเตอร์ โดยการใช้ฟังก์ชันระบบของเอมเอสดอส หมายเลขฟังก์ชัน 25h ทำหน้าที่กำหนดค่าใหม่ให้กับหมายเลขอินเทอร์รัปต์เวคเตอร์นั้น จากการระบุค่าในรีจิสเตอร์ DS และ DX ลักษณะการใช้ฟังก์ชัน 25h เป็นดังนี้

```
 :          :
mov     DS,old_segment   ; restore old interrupt vector
mov     DX,old_offset
mov     AL,Vect_Num      ; restore to Vector Number
mov     AH,25h           ; set interrupt vector
int     21h
 :          :
```

2. ปล่อยเนื้อที่หน่วยความจำที่ติดตั้งฟังก์ชันเหล่านั้น โดยการใช้ฟังก์ชันระบบของเอมเอสดอส ทำหน้าที่ปล่อยเนื้อที่หน่วยความจำ ที่ตำแหน่งเซกเมนต์ซึ่งเป็นตำแหน่งของพีเอสพีของโปรแกรมที่บรรจุในหน่วยความจำส่วนนั้น