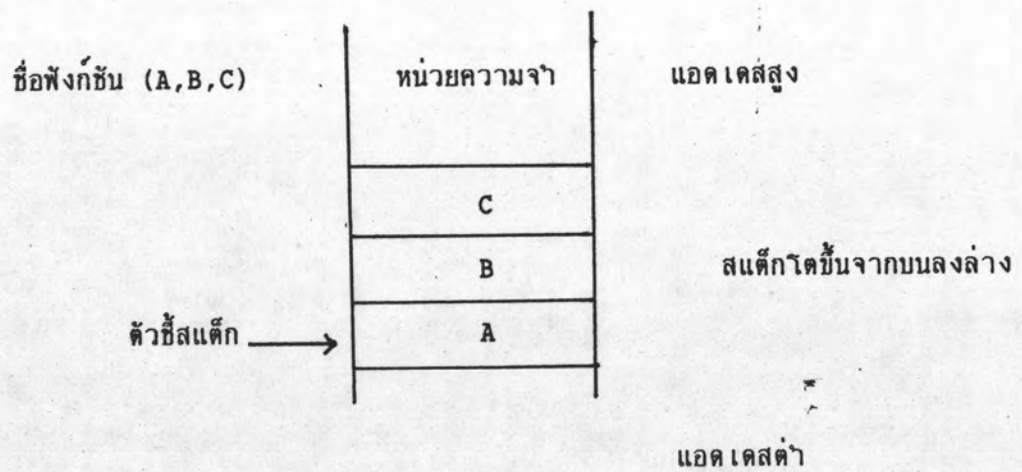
บทที่ 3

การส่งผ่านพารามิเตอร์ระหว่างโปรแกรมบนไมโครคอมพิวเตอร์

ในบทนี้จะกล่าวถึง การติดต่อกันระหว่างโปรแกรมภาษา ซี โคบอล เบสิค หรือ แอสเซมบลีกับโปรแกรมย่อยภาษาแอสเซมบลี

3.1 การใช้ภาษาซีกับภาษาแอสเซมบลี [9] [11]

การเขียนโปรแกรมย่อยภาษาแอสเซมบลี เพื่อให้โปรแกรมภาษาซีเรียกใช้ จะต้องสร้างโปรแกรมภาษาแอสเซมบลีในลักษณะของฟังก์ชัน เมื่อโปรแกรมภาษาซีเรียกใช้ฟังก์ชันใดและต้องการส่งพารามิเตอร์ไปยังโปรแกรมย่อย จะต้องกำหนดพารามิเตอร์ในวงเล็บหลังฟังก์ชันซึ่งพารามิเตอร์เหล่านี้จะถูกเก็บลงในสแต็กจากขวามาซ้ายดัง รูปที่ 3.1 และค่าของตัวชี้สแต็กจะถูกลดค่าเท่ากับจำนวนของพารามิเตอร์ คูณกับจำนวนของเนื้อที่ที่ใช้เก็บแอดเดสของพารามิเตอร์



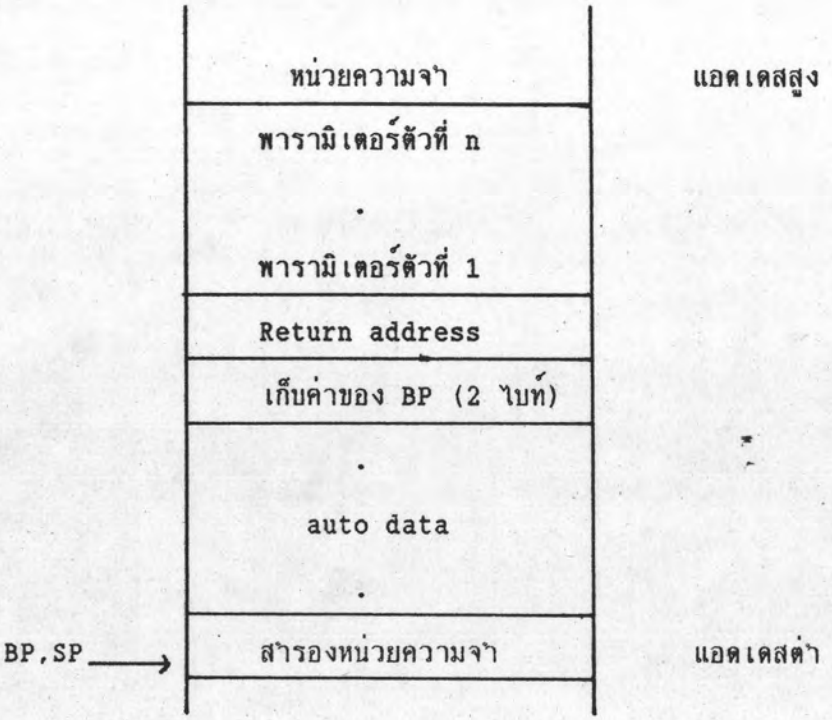
รูปที่ 3.1 แสดงการทำงานของคอมพิวเตอร์ เมื่อโปรแกรมมีการเรียกใช้ฟังก์ชัน คอมพิวเตอร์จะทำการเก็บค่าของพารามิเตอร์ C B และ A ลงในสแต็กตามลำดับและตัวชี้สแต็ก (Stack Pointer) จะชี้ที่ตำแหน่งของ A

ขั้นตอนสำคัญๆ ที่ฟังก์ชัน (โปรแกรมย่อย) จะต้องกระทำในช่วงแรกๆ ก่อนการทำงานในรายละเอียดของฟังก์ชันนั้นจะเป็นดังนี้

1. มีคำสั่งที่จะทำการเก็บค่าของรีจิสเตอร์ BP ลงในสแต็ก (PUSH BP)

2. ค่าของตัวชี้สแต็กถูกลดค่าลง โดยค่าของจำนวนไบท์ที่จะต้องใช้เก็บค่าของรีจิสเตอร์ BP (ลดลงจำนวน 2 ไบท์) นอกจากนี้ในสแต็กอาจจะมีความจำเป็นในการเก็บข้อมูลชนิด auto ที่ได้กำหนดไว้ภายในฟังก์ชันและยังต้องมีเนื้อที่สำรองของหน่วยความจำซึ่งอาจจะใช้เมื่อมีการประมวลผลนิพจน์ต่างๆ อีกด้วย แต่ถ้าไม่มีการกำหนดตัวแปร auto หรือไม่มีความจำเป็นในการสำรองเนื้อที่ตรงหน่วยความจำแล้ว ค่าของ SP ก็จะไม่เปลี่ยนแปลง

3. ทำการเก็บค่าของตัวชี้แสต็ก SP ไว้ในรีจิสเตอร์ BP เพื่อที่จะทำให้มีการอ่านค่าจากหน่วยความจำสแต็ก (ไม่ว่าจะเป็นค่าของอาร์กิวเมนต์ของฟังก์ชั่น หรือตัวแปรชนิด auto) และหน่วยความจำสำรองได้สะดวก โดยใช้ค่าในรีจิสเตอร์ BP เป็นตัวชี้หรือตัวอ้างอิง



รูปที่ 3.2 แสดงการจัดเนื้อที่ในสแต็ก หลังจากได้เริ่มต้นทำงานตามฟังก์ชันในช่วงแรกแล้ว

โดยปรกติแล้วค่าของ BP และ SP ภายในการทำงานของฟังก์ชันนั้นจะมีค่าเท่ากันตลอด ให้พิจารณาการจัดการลำดับต่าง ๆ ภายในหน่วยความจำเมื่อมีการเรียกใช้ฟังก์ชันแล้วดังรูปที่ 3.2

เมื่อฟังก์ชันได้ถูกประมวลผล หรือได้ถูกเอ็กซีคิวต์ตามที่ได้โปรแกรมไว้แล้ว สิ่งที่เราควรทราบอีกประการหนึ่งก็คือ ขั้นตอนหรือวิธีการส่งค่าคำตอบกลับ (Return value) ไปยังโปรแกรมหลักที่เรียกใช้ฟังก์ชันตามที่กำหนด (บางฟังก์ชันอาจไม่มี return value ก็ได้) ฟังก์ชันที่มี return value ได้นั้นจะต้องกำหนดชนิด หรือขนาดของข้อมูลของฟังก์ชันนั้นด้วยคำสั่งในตอนท้ายของฟังก์ชันนั้นโดยจะส่งกลับไว้ในรีจิสเตอร์ที่กำหนด ดังนี้

ข้อมูล 16 บิต จะส่งข้อมูลกลับในรีจิสเตอร์ AX

ข้อมูล 32 บิท จะส่งข้อมูลกลับในรีจิสเตอร์ (AX,BX)

ข้อมูล 64 บิท จะส่งข้อมูลกลับในรีจิสเตอร์ (AX,BX,CX,DX)

การทำ return value ขึ้นอยู่กับการกำหนดชนิดของข้อมูลของฟังก์ชันเอง เช่น ถ้าฟังก์ชันถูกกำหนดเป็น long int ค่าของ return value จะเป็นขนาด 32 บิท และถูกเก็บไว้ในรีจิสเตอร์ AX และ BX (โดยที่ AX เก็บค่าที่นัยสำคัญสูงสุด) เป็นต้น

หลังจากที่ฟังก์ชันได้ทำการกำหนด return value ให้กับรีจิสเตอร์ที่ต้องการแล้วจะมีคำสั่งในการอ่านค่า BP ของเดิม (ก่อนการเรียกฟังก์ชัน) กลับมา (POP BP) ส่วนคำสั่งสุดท้ายของฟังก์ชันก็คือ คำสั่ง RET นั่นเอง

ในการเขียนโปรแกรมด้วยภาษาแอสเซมบลี ให้มีความสามารถในการเชื่อมโยงกับโปรแกรมภาษาชีนั้น ก็อาศัยหลักการในการเขียนให้เป็นฟังก์ชันใหม่ฟังก์ชันหนึ่ง โดยให้มีคุณสมบัติสำคัญดังที่กล่าวมาแล้ว สำหรับโปรแกรมภาษาแอสเซมบลีที่เขียนขึ้นนี้จะต้องมีรูปแบบของโปรแกรมด้วยคำสั่งดังต่อไปนี้

```
<text>      SEGMENT     BYTE PUBLIC 'CODE'
            ASSUME      CS:<text>, DS:<dseg>
              .
            <..... code segment .....>
              .
<text>      ENDS

<dseg>      GROUP       _DATA,_BSS
<data>      SEGMENT     WORD PUBLIC 'DATA'
              .
              .
            <...initialized data segment ...>
              .
              .
<data>      ENDS

_BSS        SEGMENT     WORD PUBLIC 'BSS'
              .
            <...uninitialized data segment..>
              .
_BSS        ENDS

            END
```

สำหรับการกำหนด ‹text› ‹dseg› และ ‹data› มีวิธีการใช้งานตามแต่ละชนิดของแบบของหน่วยความจำที่ใช้งานดังตารางที่ 3.1

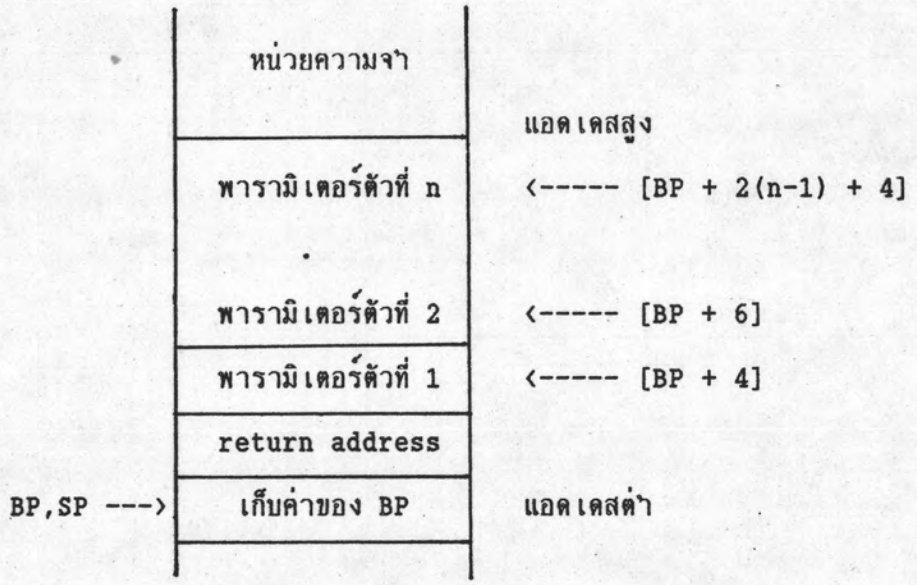
Model	Identifier Replacements	Code and Data Pointer
Tiny,Small	‹code›=_TEXT ‹data›=_DATA ‹dseg›=_DGROUP	Code: DW _TEXT:xxx Data: DW DGROUP:xxx
Compact	‹code›=_TEXT ‹data›=_DATA ‹dseg›=_DGROUP	Code: DW _TEXT:xxx Data: DW DGROUP:xxx
Medium	‹code›=filename_TEXT ‹data›=_DATA ‹dseg›=DGROUP	Code: DD xxx Data: DW DGROUP:xxx
Large	‹code›=filename_TEXT ‹data›=_DATA ‹dseg›=DGROUP	Code: DD xxx Data: DD DGROUP:xxx
Huge	‹code›=filename_TEXT ‹data›=filename_DATA ‹dseg›=filename_DATA	Code: DD xxx Data: DD xxx

ตารางที่ 3.1 การแทนค่าของ Identifier ตามชนิดของหน่วยความจำที่ใช้งาน

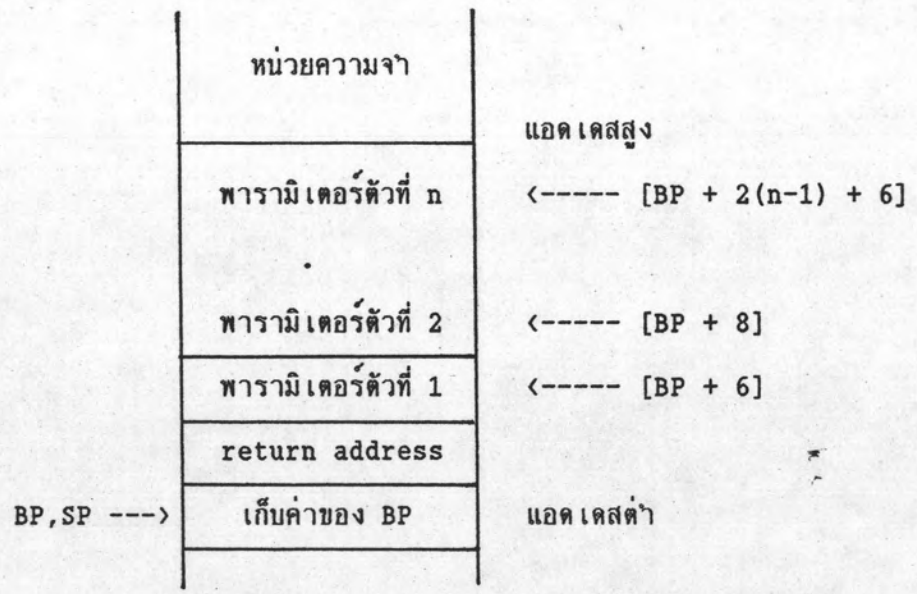
แล้วตามด้วยชื่อของฟังก์ชัน (อาจจะเขียนหลายฟังก์ชันในโปรแกรมเดียวกันได้) ซึ่งจะต้องประกาศไว้ก่อนด้วยคำสั่ง PUBLIC ต่อไปนี้ (สมมติตั้งชื่อฟังก์ชันว่า FUNC)

```
        PUBLIC   FUNC
          .
          .
FUNC    PROC     NEAR (หรือใช้ FAR ก็ได้)
        PUSH     BP
        MOV      BP,SP
          .
          .
        ตัวโปรแกรม
          .
          .
        POP      BP
        RET
FUNC    ENDP
```

สำหรับโปรแกรมภาษาแอสเซมบลีนั้น ในการอ้างอิงถึงพารามิเตอร์ที่ส่งมาจากภาษาซีขึ้นอยู่กับการกำหนดว่าจะเป็นการเรียกในระยะใกล้ (ใช้ NEAR) หรือเป็นการเรียกในระยะไกล (ใช้ FAR) โดยมีลักษณะการอ้างอิงแตกต่างกันดังรูปที่ 3.3 และ 3.4 และในรูปที่ 3.5 จะเป็นตัวอย่างโปรแกรมหลักภาษาซีที่มีการเรียกใช้ฟังก์ชันที่เขียนด้วยภาษาแอสเซมบลีดังในรูปที่ 3.6



รูปที่ 3.3 แสดงการอ้างอิงถึงพารามิเตอร์ต่าง ๆ ที่กำหนดเป็นการเรียกในระยะสั้น



รูปที่ 3.4 แสดงการอ้างอิงถึงพารามิเตอร์ต่าง ๆ ที่กำหนดการเรียกในระยะไกล

```
/*                                                              */
/*  This is sample main program to call assembly subroutine     */
/*                                                              */

#define    BEEP    '\007'
main()
{
     int  choice = 0;
     while  (choice != '5')
        {
        cls();
        choice = displaymenu();
        switch (choice)
            {
            case  '1': scrollup();
                       break;
            case  '2': scrolldown();
                       break;
            case  '3': setcursor();
                       break;
            case  '4': readcursor();
                       break;
            default:   beeping();
            }
       }
}
int displaymenu()
{
    int  ch;
    setcur( 4,20);printf("                MENU\n");
    setcur(10,20);printf("1.  Scroll up this menu 3 lines\n");
    setcur(11,20);printf("2.  Scroll down this menu 2 lines\n");
    setcur(12,20);printf("3.  Set cursor to coordinate (10,5)\n");
    setcur(13,20);printf("4.  Read current cursor position\n");
    setcur(14,20);printf("5.  Exit\n");
    setcur(20,20);ch = getch();
    putch(ch);
    return(ch);
}
```

รูปที่ 3.5 แสดงตัวอย่างโปรแกรมภาษาซี

```
scrollup()
{
    pageup(7,0,17,79,3);
    getch();
}

scrolldown()
{
    pagedown(7,0,17,79,2);
    getch();
}

setcursor()
{
    setcur(10,5);
    getch();
}

readcursor()
{
    int  x,y;
    readcur(&x,&y);
    printf("\n\nThe position of the cursor y,x is :  (%d,%d)\n",y,x);
    setcur(y,x);
    getch();
}

beeping()
{
    putch(BEEP);
}
```

รูปที่ 3.5 แสดงตัวอย่างโปรแกรมภาษาซี (ต่อ)

```
;*****************************************************************
;*                                                               *
;*  This is sample ASSEMBLY subroutine called by TURBOC (main)  *
;*                                                               *
;*****************************************************************

_data       segment   byte public 'data'
_data       ends
_bss        segment   byte public 'bss'
_bss        ends

dgroup      group     _data, _bss

_text       SEGMENT   BYTE PUBLIC 'code'
            PUBLIC    _CLS, _PAGEUP, _PAGEDOWN, _SETCUR, _READCUR
            ASSUME    CS:_text, ds:dgroup

_CLS        proc      near
            push      bp
            sti
            mov       ah,15
            int       10h
            mov       ah,0
            int       10h
            mov       ax,0600h
            mov       cx,0000h
            mov       dx,184fh
            mov       bh,07h
            int       10h
            mov       ah,2
            mov       dx,0
            mov       bh,0
            int       10h
            pop       bp
            ret
_CLS         endp
;
;
_PAGEUP      proc      near
            push      bp
            mov       bp,sp
            mov       cx,[bp+6]
            mov       dx,[bp+4]
            mov       ch,dl
            mov       dx,[bp+10]
            mov       bx,[bp+8]
            mov       dh,bl
            mov       ax,[bp+12]
            mov       bh,07h
            mov       ah,06h
            int       10h
            pop       bp
            ret
_PAGEUP     endp
;
;
```

รูปที่ 3.6 แสดงตัวอย่างโปรแกรมย่อยภาษาแอสเซมบลีสำหรับโปรแกรมภาษาซี

```
_PAGEDOWN   proc        near
            push        bp
            mov         bp,sp
            mov         cx,[bp+6]
            mov         dx,[bp+4]
            mov         ch,dl
            mov         dx,[bp+10]
            mov         bx,[bp+8]
            mov         dh,bl
            mov         ax,[bp+12]
            mov         bh,07h
            mov         ah,07h
            int         10h
            ret
_PAGEDOWN   endp
;
;
_SETCUR     proc        near
            push        bp
            mov         bp,sp
            mov         ah,15
            int         10h
            mov         dx,[bp+6]
            mov         cx,[bp+4]
            mov         dh,cl
            mov         ah,02h
            int         10h
            pop         bp
            ret
_SETCUR     endp
;
;
_READCUR    proc        near
            push        bp
            mov         bp,sp
            mov         ah,15
            int         10h
            mov         ah,03h
            int         10h
            mov         bx,[bp+4]
            mov         ch,0
            mov         cl,dh
            mov         [bx],cx
            mov         bx,[bp+6]
            mov         cl,dl
            mov         [bx],cx
            pop         bp
            ret
_READCUR    endp
;
_text       ENDS
            END
```

รูปที่ 3.6 แสดงตัวอย่างโปรแกรมย่อยภาษาแอสเซมบลีสำหรับโปรแกรมภาษาซี (ต่อ)

3.2 การใช้ภาษาโคบอลกับภาษาแอสเซมบลี

ภาษาโคบอลที่ใช้สำหรับวิทยานิพนธ์นี้ใช้ของไมโครซอฟต์ ซึ่งวิธีการเรียกใช้โปรแกรมย่อยในภาษาโคบอลแบ่งออกเป็น 4 แบบคือ

1. เป็นการถ่ายโอนการทำงานชั่วคราวไปยังโปรแกรมย่อยภาษาโคบอล โดยใช้คำสั่ง CALL
2. เป็นการถ่ายโอนการทำงานชั่วคราวไปยังโปรแกรมย่อยที่ไม่ใช่ภาษาโคบอล โดยการใช้คำสั่ง CALL
3. เป็นการถ่ายโอนการทำงานถาวรไปยังโปรแกรมย่อยภาษาโคบอล โดยใช้คำสั่ง CHAIN
4. เป็นการถ่ายโอนการทำงานถาวรไปยังโปรแกรมย่อยที่ไม่ใช่ภาษาโคบอล โดยใช้คำสั่ง CHAIN

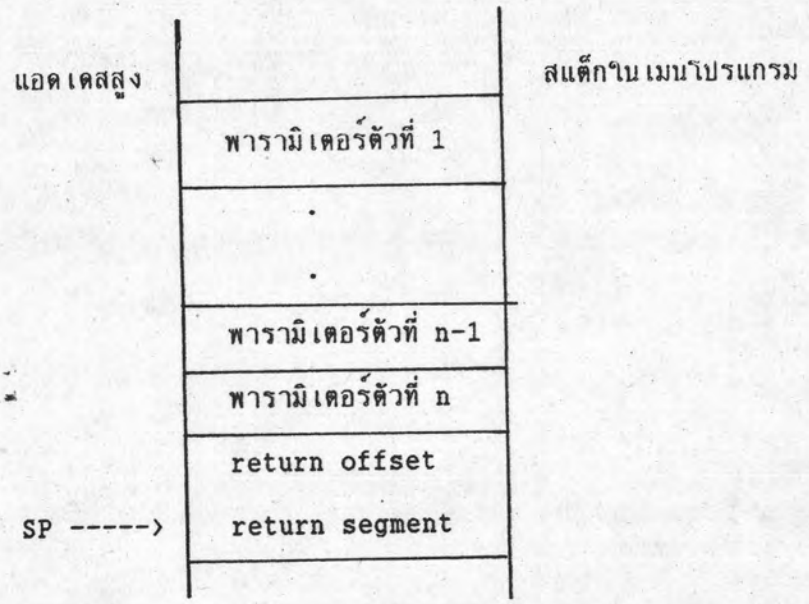
ในวิทยานิพนธ์ส่วนนี้จะกล่าวเฉพาะในข้อ 2 เท่านั้น และในภาษาโคบอลนั้นการส่งข้อมูลระหว่างโปรแกรมนั้นเราจะกำหนดโดยการใช้ USING ในคำสั่ง CALL สำหรับรูปแบบของคำสั่งในการเรียกโปรแกรมย่อยในภาษาโคบอลมีรายละเอียดดังนี้

```
CALL {literal-1 | identifier-1}
     [USING data-name-1 [, data-name-2]...]
```

โดยที่ literal-1 หรือ identifier-1 จะเป็นชื่อของโปรแกรมย่อย

data-name-1 เป็นข้อมูลที่จะส่งไปยังโปรแกรมย่อย

สำหรับโปรแกรมย่อยที่เขียนนั้นจะถูกเรียกแบบระยะไกล (Far Call) จากโปรแกรมภาษาโคบอลเท่านั้น และการส่งผ่านข้อมูลหรือพารามิเตอร์ระหว่างโปรแกรมจะทำโดยการเก็บแอดเดสของพารามิเตอร์แต่ละตัวลงสแต็ก โดยพารามิเตอร์ตัวแรกจะถูกเก็บในสแต็กก่อนและตัวถัดไปก็จะเก็บลงไปตามลำดับดังรูปที่ 3.7 ซึ่งแสดงรายละเอียดการเก็บข้อมูลต่าง ๆ ลงสแต็กก่อนโอนการทำงานให้โปรแกรมย่อย



รูปที่ 3.7 แสดงข้อมูลที่เก็บในสแต็กก่อนโอนการทำงานให้โปรแกรมย่อย

สำหรับชื่อของโปรแกรมย่อยภาษาแอสเซมบลีนั้น จะต้องกำหนดด้วย PUBLIC และประกาศว่าโปรแกรมย่อยนี้จะถูกเรียกใช้แบบระยะไกลโดยบอกใน PROC FAR ในภาษาโคบอลของไมโครซอฟต์นี้จะมีแมโคร (macro) มาให้สำหรับการเขียนโปรแกรมย่อยภาษาแอสเซมบลี โดยมีแมโคร START CSEG, END CSEG, START DSEG และ END DSEG ซึ่งแมโครเหล่านี้จะเก็บไว้ในแฟ้มข้อมูลชื่อ USERSEG.mAC ดังนั้นโปรแกรมย่อยที่เขียนขึ้นมาจะต้องมีการอ้างถึงแฟ้มข้อมูลนี้ด้วย โดยการใช้คำสั่ง INCLUDE filename ใส่ในโปรแกรมด้วย ในการตั้งชื่อของโปรแกรมย่อยนั้นจะต้องไม่ตั้งชื่อให้ตรงกับชื่อของมอดูล (module) ของตัวปฏิบัติการเวลาดำเนินโปรแกรม (runtime executor) ของไมโครซอฟต์โคบอล และการใช้สแต็กของโปรแกรมย่อยจะต้องไม่ทำให้เกิดการล้นสแต็ก (stack overflow) หรือเกิดการน้อยเกินไปของสแต็ก (stack underflow) ดังนั้นถ้าไม่แน่ใจในการใช้สแต็กก็ไม่ควรใช้สแต็กของโคบอล ควรจะสร้างสแต็กในโปรแกรมขึ้นมาใช้แยกต่างหาก

สำหรับซอฟต์แวร์ของไมโครซอฟต์โคบอลที่จะนำมาเชื่อมโยง (link) กับภาษาแอสเซมบลีนั้นจะมีขั้นตอนในการใช้งานดังนี้

1. ตรวจสอบว่าซอฟต์แวร์ที่มีนั้นประกอบด้วยแฟ้มข้อมูล เหล่านี้

COBOL1.LIB	PSEG.MAC
COBOL2.LIB	INSTALL.COM
DEBUG.LIB	INSTALL.MSG
COBOL1.OBJ	INSTALL.OVL
COBOL2.OBJ	INSTALL.SPC
DEBUG.OBJ	INSTALL.DAT
ASM.ASM	MAKERUN.BAT
USERSEG.MAC	MAKERUN2.BAT
USERPROG.MAC	LINK.EXE

2. ใช้โปรแกรมบรรณาธิการ (Editor) เช่น EDLIN หรือ SideKick เป็นต้น สำหรับการแก้ไขข้อมูลในแฟ้มข้อมูล USERPROG.MAC เพื่อบอกว่ามีโปรแกรมย่อยภาษาแอสเซมบลีอะไรที่มีการใช้ร่วมกับโปรแกรมภาษาโคบอล โดยกำหนดในคำสั่งดังข้างล่างนี้

```
ASMNAM subroutine-entry-point-name,type
```

โดย type จะเป็นชนิดของโปรแกรมย่อยที่จะใช้งานมีได้ดังนี้

ASM86	สำหรับโปรแกรมย่อยภาษาแอสเซมบลี
C	สำหรับโปรแกรมย่อยภาษาซี
PASCAL	สำหรับโปรแกรมย่อยภาษาปาสกาล
FORTRAN	สำหรับโปรแกรมย่อยภาษาฟอร์แทรน

* สำหรับภาษาซี ภาษาปาสกาล และภาษาฟอร์แทรนยังไม่สามารถใช้งานได้ในซอฟต์แวร์ของเวอร์ชันนี้ ซึ่งรายละเอียดว่าใช้งานได้สำหรับภาษาอะไรบ้าง อ่านได้จากแฟ้มข้อมูลชื่อ UPDATE.DOC

3. ในโปรแกรมย่อยภาษาแอสเซมบลีจะต้องมีแฟ้มข้อมูล USERSEG.MAC ในโปรแกรมและการเริ่มการใช้กลุ่มของข้อมูลให้ขึ้นต้นด้วยแมโคร START DSEG และปิดท้ายด้วยแมโคร END DSEG ในทำนองเดียวกันในกลุ่มของคำสั่งให้ขึ้นต้นด้วย START CSEG และปิดท้ายของกลุ่มของคำสั่งด้วย END CSEG เมื่อโปรแกรมได้เขียนเสร็จแล้วก็ทำการแปลโปรแกรม (Compile)

4. ใช้แฟ้มข้อมูล MAKERUN.BAT เพื่อนำโปรแกรมย่อยภาษาแอสเซมบลีมาทำการเชื่อมโยงกันให้เป็นตัวปฏิบัติการเวลาดำเนินโปรแกรม (Runtime executor) สำหรับใช้ดำเนิน (Run) โปรแกรมภาษาโคบอล โดยมีรูปแบบการใช้คำสั่งดังนี้

```
MAKERUN runtime-executor-name object-file-1
        object-file-2 ... object-file-8
```

โดย	runtime-executor-name	เป็นชื่อของแฟ้มข้อมูลที่สร้างขึ้นมาสำหรับโปรแกรมภาษาโคบอลที่มีการเรียกใช้โปรแกรมย่อย ซึ่งมีอยู่ในมอดูลของตัวปฏิบัติการเวลาดำเนินโปรแกรม
	object-file-1 .. 8	เป็นชื่อของโปรแกรมย่อยภาษาแอสเซมบลีมีความยาว 1-8 ตัวอักษร

5. ทำการแปลโปรแกรมภาษาโคบอลจากนั้นจึงทำการดำเนิน(Run) โปรแกรมได้โดยคีย์ชื่อของตัวปฏิบัติการเวลาดำเนินโปรแกรม ตามด้วยชื่อของโปรแกรมภาษาโคบอล เช่นโปรแกรมภาษาโคบอลชื่อ COBTEST.OBJ และตัวปฏิบัติการเวลาดำเนินโปรแกรมชื่อ RUNSUB.EXE ให้ทำดังนี้

```
A> RUNSUB COBTEST
```

ในส่วนของการใช้โปรแกรมภาษาโคบอลติดต่อกับโปรแกรมภาษาแอสเซมบลี จะแสดงตัวอย่างการใช้งานและการเขียนโปรแกรมโดยแบ่งขั้นตอนเป็น 6 ขั้นตอน โดยตัวอย่างนี้ต้องการให้โปรแกรมภาษาโคบอลส่งค่าของตัวเลข 2 ตัว ไปลบกันในโปรแกรมย่อยภาษาแอสเซมบลีและส่งผลลัพท์กลับมาในตัวแปรตัวที่ 3 โดยมีขั้นตอนต่าง ๆ ดังนี้

1. สร้างโปรแกรมภาษาโคบอล

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CALL1.
ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
01  WORK-AREAS.
    05  PARM1     PIC S9(5) COMP-0 VALUE 511.
    05  PARM2     PIC S9(5) COMP-0 VALUE 384.
    05  PARM3     PIC S9(5) COMP-0 VALUE ZERO.
    05  PPARM-1   PIC -----9 VALUE ZERO.
    05  PPARM-2   PIC -----9 VALUE ZERO.
    05  PPARM-3   PIC -----9 VALUE ZERO.
PROCEDURE DIVISION.
MAIN.
    CALL "SUBIT" USING PARM1, PARM2, PARM3.
    MOVE PARM1 TO PPARM-1.
    MOVE PARM2 TO PPARM-2.
    MOVE PARM3 TO PPARM-3.
    DISPLAY PPARM-1 " - " PPARM-2 " = " PPARM-3.
    STOP RUN.
```

2. แก้ไขค่าในแฟ้มข้อมูล USERPROG.MAC ดังนี้

```
asmnam SUBIT,ASM86
```

3. สร้างโปรแกรมย่อยภาษาแอสเซมบลี โดยมีส่วนของโปรแกรมดังนี้

```
        include USERSEG.MAC
        PAGE
; Structure for accessing parameters on the stack
DYNS    STRUC
        DW      ?
        DD      ?
PARM3   DW      ?
PARM2   DW      ?
PARM1   DW      ?
DYNS    ENDS
;
START_CSEG SUBIT
        push    bp
        mov     bp,sp
        mov     bx,[bp].parm1    ; get addr. of parm1
        mov     ax,[bx]
        xchg    ah,al   ; COBOL COMP-0 bytes reversed
        mov     bx,[bp].parm2    ; get addr. of parm2
        mov     cx,[bx]
        xchg    ch,cl   ; COBOL COMP-0 bytes reversed
        sub     ax,cx   ; parm1 - parm2
        xchg    ah,al  ; restore byte order of result
        mov     di,[bp].parm3
```

```
        mov     [di],ax
        pop     bp
        ret         ; note COBOL will pop arguments
                    ; off stack, but it can also be
                    ; done here using ret 6 will no
                    ; bad results
END_CSEG SUBIT
        END
```

4. ทำการแปลโปรแกรมภาษาโคบอล โดยให้ชื่อของโปรแกรมที่แปลแล้วมีชื่อ CALL1.OBJ และทำการแปลโปรแกรมภาษาแอสเซมบลีโดยให้ชื่อว่า SUBIT.OBJ

5. สร้างตัวปฏิบัติการเวลาดำเนินโปรแกรมโดยให้ชื่อว่า RUN1.EXE โดยทำดังนี้

```
A> MAKERUN RUN1 SUBIT
```

โดย RUN1 คือชื่อของตัวปฏิบัติการเวลาดำเนินโปรแกรมที่ต้องการสร้าง

SUBIT คือชื่อของรหัสประสงค์ (Object code) ของโปรแกรมภาษาแอสเซมบลีที่สร้างสำหรับโปรแกรมภาษาโคบอล

6. นำ RUN1 มาใช้สำการดำเนินโปรแกรม(run) ภาษาโคบอล CALL1 โดยการคีย์คำสั่งดังนี้

```
A> RUN1 CALL1
```

3.3 การใช้ภาษา เบสิคกับภาษาแอส เซมบลี [10]

ภาษา เบสิคที่ใช้สำหรับวิทยานิพธ์ฉบับนี้ใช้ เทอร์โบ เบสิค (TURBO BASIC) ซึ่งการเรียกใช้โปรแกรมย่อยแบ่งออกเป็น 3 ชนิดคือ

1. ใช้คำสั่ง CALL เรียกไปยังโพรซีเยอร์ซึ่งต้องการให้ทำงานและโพรซีเยอร์นั้นจะเป็นโปรแกรมที่เขียนด้วยภาษาแอส เซมบลีโดยสร้างในแฟ้มข้อมูล INLINE.COM หรือเป็นภาษาแอส เซมบลีที่เขียนในภาษา เบสิคในรูปแบบคำสั่งของ $INLINE

2. ใช้คำสั่ง CALL ABSOLUTE ไปยังแอดเดสของโปรแกรมแอส เซมบลีที่เขียนขึ้น

3. ใช้คำสั่ง CALL INTERUPT โดยสามารถใช้ได้ในลักษณะ เดียวกับการใช้ซอฟต์แวร์อิน เทอร์รัพท์ของโปรแกรมภาษาแอส เซมบลี

สำหรับวิทยานิพนธ์ฉบับนี้จะกล่าว เฉพาะในข้อที่ 1 เพราะ เป็นหลักในการ เขียนโปรแกรม เชื่อมโยงภาษาแอส เซมบลี รูปแบบของการ เรียกใช้โปรแกรมย่อยแบบ INLINE จะมีรูปแบบการใช้งานดังนี้

```
.
CALL <procname> [(parameter list)]
.
SUB <procname> INLINE
    $INLINE "COM filename"
END SUB
```

โดยที่ procname จะ เป็นชื่อของโพรซีเยอร์ที่ใช้สำหรับกำหนดว่าต้องการ เรียกใช้โปรแกรมย่อยภาษาแอส เซมบลีอะไร

COM filename จะ เป็นชื่อโปรแกรมย่อยภาษาแอส เซมบลีที่จัดการให้เป็นแฟ้มข้อมูลแบบ COM แล้ว

การสร้างแฟ้มข้อมูลแบบ INLINE.COM มีขั้นตอนในการจัดทำจากโปรแกรมภาษาแอสเซมบลีดังนี้

1. ใช้ MASM สำหรับการแปลโปรแกรมภาษาแอสเซมบลี
2. ใช้ LINK สำหรับโปรแกรมภาษาแอสเซมบลีที่ผ่านการแปลแล้ว
3. ใช้ EXE2BIN สำหรับการเปลี่ยนแฟ้มข้อมูลแบบ .EXE เป็น .COM

สำหรับโปรแกรมภาษาแอสเซมบลีที่ต้องการใช้แบบ .COM นั้นมีข้อกำหนดในการใช้งานดังนี้

1. ในโปรแกรมภาษาแอสเซมบลีที่เขียนจะต้องไม่มีส่วนของสแต็ก (Stack Segment)
2. ใช้ ORG 100H ในตอนเริ่มต้นของโปรแกรม
3. การอ้างอิงถึงข้อมูลและคำสั่งต่าง ๆ ให้ทำในลักษณะของการอ้างอิงแบบตำแหน่งที่อยู่สัมพัทธ์

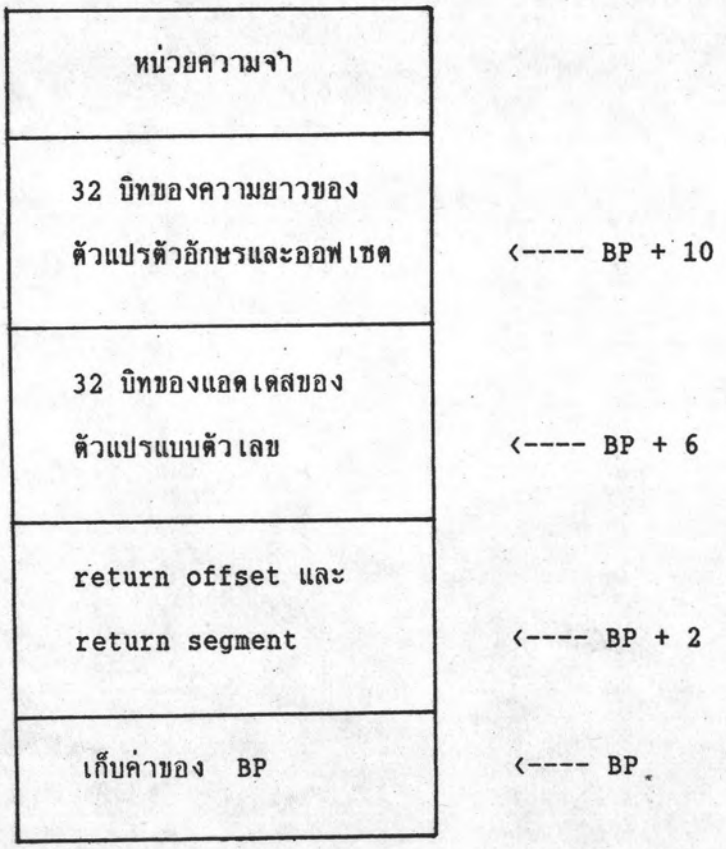
การส่งผ่านพารามิเตอร์ในภาษาเบสิคไปยังโปรแกรมย่อย แบ่งแยกได้หลายแบบตามชนิดของตัวแปรที่ใช้สำหรับวิทยานิพนธ์นี้จะกล่าวเฉพาะตัวแปรแบบตัวเลข (Numeric Variable) และตัวแปรแบบตัวอักษร (String Variable)

1. การส่งผ่านของตัวแปรตัวเลข กระทำโดยการส่งแอดเดสของตัวแปรขนาด 32 บิทไปยังโปรแกรมย่อย และตัวแปรตัวเลขแบบจำนวนเต็มจะใช้เนื้อที่ขนาด 16 บิทโดยมีการจัดเรียงค่าให้ไบท์ที่มีนัยสำคัญน้อยกว่าเก็บที่แอดเดสในหน่วยความจำที่ต่ำกว่า เช่น 513 มีค่าเท่ากับ 213H จะมีการจัดเรียงดังนี้

	byte 0	byte 1
Hex	13H	02H
Binary	00010011	00000010

2. การส่งผ่านของตัวแปรแบบตัวอักษร กระทำโดยการส่งค่าขนาด 32 บิทซึ่ง 2 ไบท์แรกจะเป็นความยาวของตัวแปรตัวอักษร และ 2 ไบท์หลังจะเป็นออฟเซตของข้อมูลในส่วนของตัวแปรแบบตัวอักษร (String segment)

ในรูปที่ 3.8 จะแสดงค่าที่เก็บในสแต็กหลังจากที่มีการเก็บค่าของรีจิสเตอร์ BP แล้วโดยสมมติว่ามีข้อมูลที่ต้องการส่งไปยังโปรแกรมย่อยภาษาแอสเซมบลี 2 ตัว เป็นตัวแปรแบบตัวเลขและตัวอักษรตามลำดับ



รูปที่ 3.8 แสดงค่าที่เก็บในสแต็กหลังจากเก็บค่าของ BP แล้ว

3.4 การใช้ภาษาแอสเซมบลีกับโปรแกรมย่อยภาษาแอสเซมบลี [8]

ในภาษาแอสเซมบลีนั้นเราสามารถเรียกโปรแกรมย่อยได้ใน 2 ลักษณะคือ

1. การเรียกโปรแกรมย่อยที่อยู่ในโปรแกรมนั้น (Intrasegment CALL) ซึ่งการเรียกอาจเป็นการเรียกในระยะใกล้คืออยู่ในช่วง +127 หรือ -128 ไบท์ หรือถ้ามีค่ามากกว่านี้ก็จะเป็นการเรียกในระยะไกล

2. การเรียกโปรแกรมที่อยู่นอก Code Segment ซึ่งเรียกว่า Intersegment CALL และโปรแกรมเชื่อมโยงสำหรับภาษาแอสเซมบลีที่สร้างขึ้นก็จะอยู่ในรูปแบบของข้อนี้

ในวิทยานิพนธ์จะกล่าวเฉพาะหัวข้อที่ 2 โดยลักษณะรูปแบบของการใช้โปรแกรมหลักเรียกโปรแกรมย่อยในภาษาแอสเซมบลีก็จะมีรูปแบบดังนี้

```
                EXTRN SUBPROG:FAR
MAINPROG:
                .
                CALL SUBPROG
                .
-------------------------------------------------

                PUBLIC SUBPROG
SUBPROG:        .
                .
                RET
-------------------------------------------------
```

คำสั่ง EXTRN จะบอกให้แอสเซมเบลอร์รู้ว่าการใช้คำสั่ง CALL ในโปรแกรมหลัก MAINPROG จะเรียกโปรแกรมย่อย SUBPROG ที่อยู่นอกเซกเมนท์นี้โดยการประกาศในคำสั่ง EXTRN ด้วยพารามิเตอร์ FAR ซึ่งในขณะที่ทำการแปลโปรแกรมแอสเซมเบลอร์ไม่สามารถรู้ได้ว่าแอดเดสของโปรแกรม SUBPROG มีค่าเป็นเท่าใด แต่แอสเซมเบลอร์จะจะใส่รหัสว่าง ๆ ก่อนและให้ LINKER ทำหน้าที่ใส่ค่าที่ถูกต้องในเวลาเชื่อมโยงโปรแกรมกัน สำหรับโปรแกรมย่อย SUBPROG จะต้องมีคำสั่ง PUBLIC ที่จะบอกให้แอสเซมเบลอร์และ LINKER รู้ถึงแอดเดสของโปรแกรมย่อย และเพื่อให้โปรแกรมสามารถทำงานได้จะต้องทำการแปลโปรแกรมทั้งสอง จากนั้นจึงนำโปรแกรมนั้นมาทำการเชื่อมโยงกันดังนี้

LINK Prompt	Reply
Object Modules [.OBJ]	B:MAINPROG+B:SUBPROG
Run File [filespec.EXE]	B:COMBPROG (หรือใช้ชื่ออื่นก็ได้)
List File [NUL.MAP]	CON
Libraries [.LIB]	[return]

ในรูปที่ 3.9 จะเป็นตัวอย่างการเขียนโปรแกรมภาษาแอสเซมบลี เพื่อเรียกโปรแกรมย่อยทำการคำนวณหาค่าของจำนวนสินค้าและราคาสินค้า

```
            page    60,132
;
TITLE       CALLMUL1 - Call Subprogram to multiply
;
            EXTRN   SUBMUL:FAR

; ---------------------------------------------------------------------
STACKSG     SEGMENT PARA STACK 'STACK'
            DW      64 DUP (?)
STACKSG     ENDS
; ---------------------------------------------------------------------
DATASG      SEGMENT PARA PUBLIC 'DATA'
QTY         DW      0140h
PRICE       DW      2500h
DATASG      ENDS
; ---------------------------------------------------------------------
CODESG      SEGMENT PARA PUBLIC 'CODE'
BEGIN       PROC    FAR
            ASSUME  CS:CODESG,DS:DATASG,SS:STACKSG
            push    ds
            sub     ax,ax
            push    ax
            mov     ax,datasg
            mov     ds,ax
            mov     ax,price        ; set up price
            mov     bx,qty          ;     & qty
            call    SUBMUL          ; call Subprogram
            ret
BEGIN       ENDP
CODESG      ENDS
            END     BEGIN


;
            page    60,132
TITLE       DOMUL1  Called subprogram, multiplies
; ---------------------------------------------------------------------
CODESG      SEGMENT PARA PUBLIC 'CODE'
SUBMUL      PROC    FAR
            ASSUME  CS:CODESG
            PUBLIC  SUBMUL
            CALL    C10MUL          ; Call subroutine
            RET
SUBMUL      ENDP
;                   Multiply routine
;                   ----------------
C10MUL      PROC    near            ; price in ax, qty in bx
            mul     bx              ; product in dx:ax
            ret
C10MUL      ENDP

CODESG      ENDS
            END     SUBMUL
```

รูปที่ 3.9 แสดงตัวอย่างโปรแกรมหลักและโปรแกรมย่อยภาษาแอสเซมบลี

การส่งผ่านพารามิเตอร์ระหว่างโปรแกรม สามารถทำโดยการส่งผ่านค่าทางสแต็กและการนำค่าไปเก็บในสแต็กกระทำโดยใช้คำสั่ง PUSH ซึ่งขั้นตอนในการส่งผ่านพารามิเตอร์ไปยังโปรแกรมย่อย โปรแกรมหลักจะมีขั้นตอนดังนี้

1. ในตอนเริ่มต้นโปรแกรมจะต้องใช้คำสั่ง PUSH DS
2. โหลดแอดเดสที่มีค่าเป็น 0 ไปเก็บในสแต็กโดยการใช้คำสั่งดังนี้

```
SUB  AX,AX
PUSH AX
```

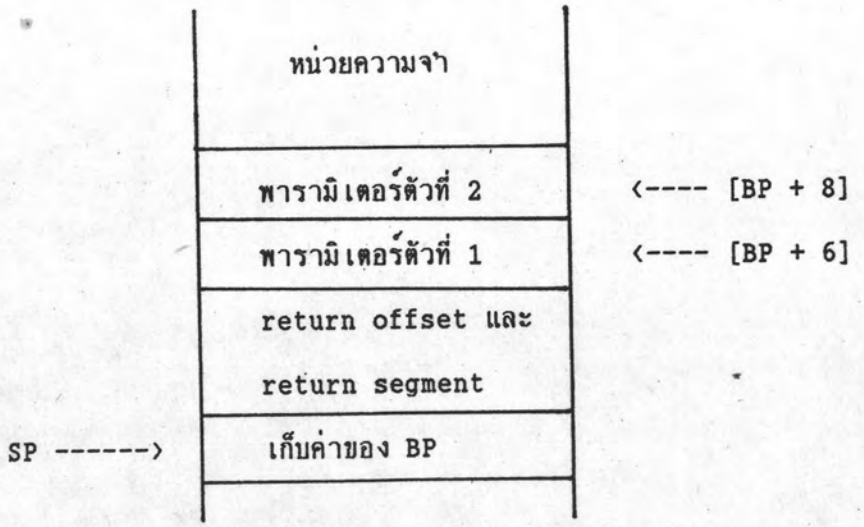
3. นำแอดเดสของตัวแปรแต่ละตัวไปเก็บในสแต็กโดยใช้คำสั่ง

```
PUSH ชื่อของตัวแปร
```

4. เรียกโปรแกรมย่อยโดยใช้คำสั่ง

```
CALL ชื่อของโปรแกรมย่อย
```

ลักษณะของสแต็กหลังจากเก็บค่าของ BP ในโปรแกรมย่อยแล้วข้อมูลที่มีอยู่ในสแต็กจะมีลักษณะดังนี้



สำหรับรูปที่ 3.10 จะแสดงการเรียกโปรแกรมย่อยพร้อมส่งผ่านพารามิเตอร์ไปยังโปรแกรมเพื่อทำการคำนวนหาค่า

```
            page    60,132
;
TITLE       CALLMUL3 - Call Subprogram, Multiplies
;
            EXTRN   SUBMUL:FAR
            PUBLIC  QTY,PRICE

; ---------------------------------------------------------------------
STACKSG     SEGMENT PARA STACK 'STACK'
            DW      64 DUP (?)
STACKSG     ENDS
; ---------------------------------------------------------------------
DATASG      SEGMENT PARA PUBLIC 'DATA'
QTY         DW      0140h
PRICE       DW      2500h
DATASG      ENDS
; ---------------------------------------------------------------------
CODESG      SEGMENT PARA PUBLIC 'CODE'
BEGIN       PROC    FAR
            ASSUME  CS:CODESG,DS:DATASG,SS:STACKSG
            push    ds
            sub     ax,ax
            push    ax
            mov     ax,datasg
            mov     ds,ax
            call    SUBMUL          ; call Subprogram
            ret
BEGIN       ENDP
CODESG      ENDS
            END     BEGIN



;
            page    60,132
TITLE       DOMUL3  Called subprogram, multiplies
            EXTRN   QTY:WORD,PRICE:WORD
; ---------------------------------------------------------------------
CODESG      SEGMENT PARA PUBLIC 'CODE'
SUBMUL      PROC    FAR
            ASSUME  CS:CODESG
            PUBLIC  SUBMUL
            CALL    C10MUL          ; Call subroutine
            RET
SUBMUL      ENDP
;                   Multiply routine
;                   ----------------
C10MUL      PROC    near
            mov     ax,price
            mov     bx,qty
            mul     bx              ; product in dx:ax
            ret
C10MUL      ENDP

CODESG      ENDS
            END     SUBMUL
```

รูปที่ 3.10 แสดงตัวอย่างเรียกใช้โปรแกรมภาษาแอสเซมบลีและส่งผ่านพารามิเตอร์ด้วย