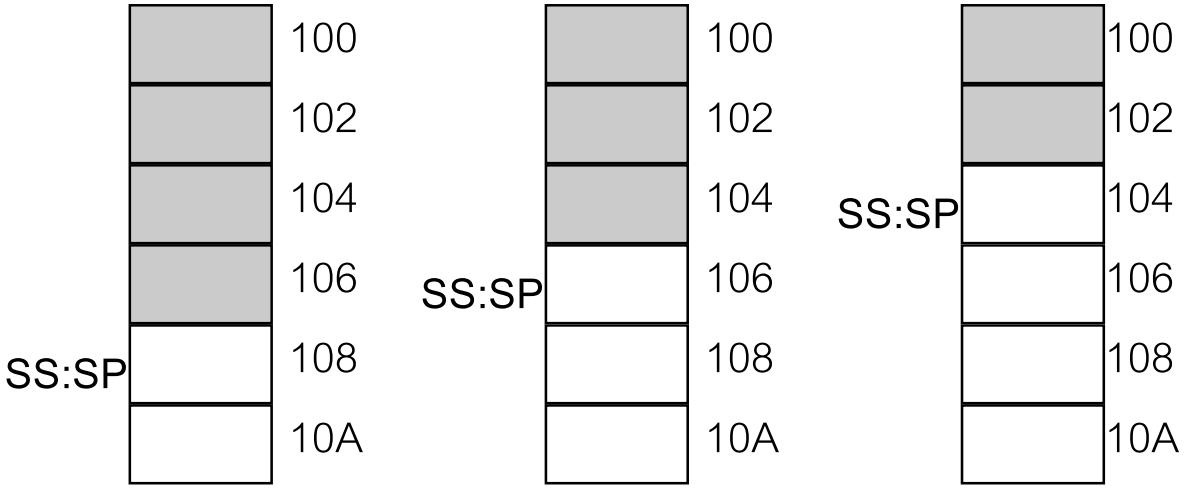
โปรแกรมย่อยและแสต็ก

- แสต็ก
- โปรแกรมย่อยที่ส่งพารามิเตอร์ผ่านทางแสต็ก
- ตัวแปรภายใน (Local variable)

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

แสต็ก

- **แสต็ก** คือหน่วยความจำที่มีลักษณะการเก็บข้อมูลแบบเก็บทีหลังอ่านออกไปก่อน (Last In First Out)
- รีจิสเตอร์ SS : SP เก็บตำแหน่งบนสุดของแสต็ก. การเก็บข้อมูลเพิ่มลงในแสต็กจะทำให้แสต็กมีทิศทางในการขยายขนาดไปทางลบ. (ค่าของรีจิสเตอร์ SP มีค่าลดลงครั้งละ 2 ไบต์)

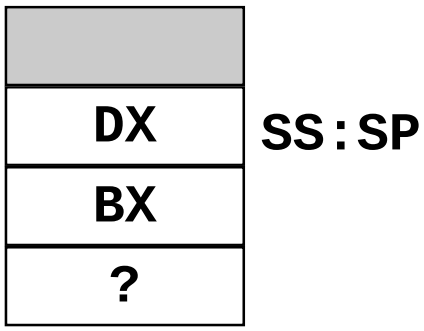


หมายเหตุ : ในการเขียนแสต็กเรานิยมเขียนข้อมูลหน่วยละ 1 เวิรด์ เพราะการเก็บข้อมูลในแสต็กจะมีขนาด 1 เวิร์ดเสมอ

แสต็ก

- **การอ้างข้อมูลในแสต็ก**
  - ใช้การอ้างแบบสัมพัทธ์กับฐาน โดยคิดอ้างอิงกับค่าในรีจิสเตอร์ BP

```
push   bx
push   dx
```




```
mov    bp,sp
mov    ax,[bp]    ;ax=dx
mov    si,[bp+2]  ;si=bx
```

  - เราสามารถใช้แสต็กในการเก็บค่าพารามิเตอร์ที่ส่งให้โปรแกรมย่อยได้

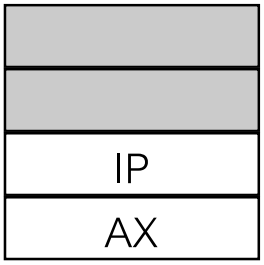
แสต็กหลังการเรียกโปรแกรมย่อย

โปรแกรมหลัก

```
push    ax
call    testproc
```

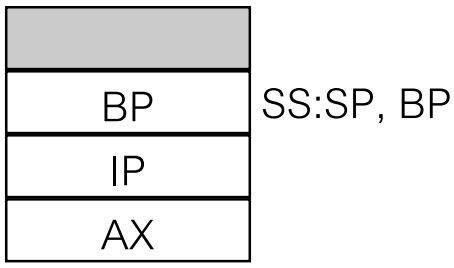
โปรแกรมย่อย

```
testproc  proc near
```



เราจะอ้างถึงข้อมูลในแสต็กโดยผ่าน BP ต้องเก็บค่าเดิมของ BP ไว้ด้วย

```
push   bp
mov    bp,sp
```



อ้างถึง AX ที่ผ่านค่ามาทางแสต็ก

```
mov    dx,[bp+4]
```

เมื่อจบโปรแกรมโปรแกรมย่อยของเราจะต้องคืนค่าให้ BP

```
pop    bp
```

และล้างข้อมูลที่ส่งมาทางแสต็กด้วย

ระบุให้ลดค่าในแสต็กไป 2 ไบต์หลังการกระโดดกลับ

```
ret    2
```

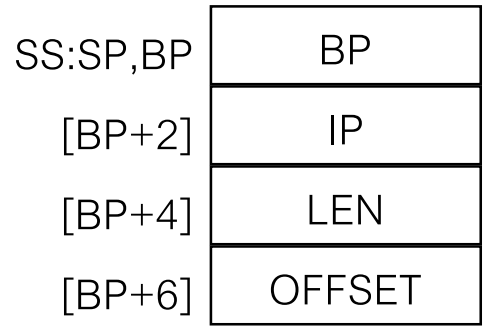
โครงสร้างทั่วไปของโปรแกรมย่อยที่ส่งค่าผ่านทางแสต็ก

```
proc1   proc  near
        push  bp
        mov   bp,sp

          ...

        pop   bp
        ret   XXX
proc1   endp
```

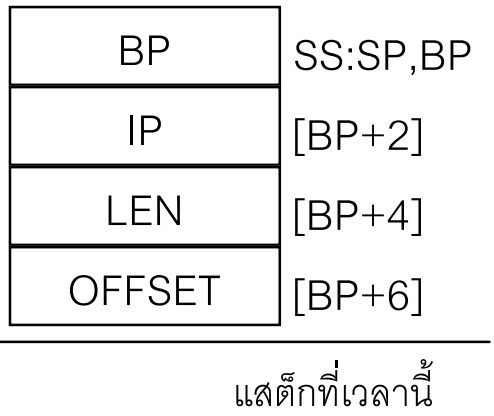
EX เขียนโปรแกรมย่อยแสดงข้อความบนหน้าจอรับพารามิเตอร์ผ่านทางแสต็ก. พารามิเตอร์มีสองค่า. พารามิเตอร์แรกคือออฟเซ็ตของข้อความมีขนาด 1 เวิรด์. พารามิเตอร์ที่สองคือความยาวของข้อความมีขนาด 1 เวิร์ดเช่นเดียวกัน. การ PUSH พารามิเตอร์ลงในแสต็กจะ PUSH พารามิเตอร์แรก ตามด้วยพารามิเตอร์ที่สอง.



ลักษณะของแสต็กหลังการเรียกใช้โปรแกรมย่อย และเก็บค่า BP ลงแสต็กเรียบร้อยแล้ว

ตัวอย่าง

```
dispstr proc  near
        push  bp
        mov   bp,sp
        push  ax
        push  bx
        push  cx
        push  dx
        mov   bx,[bp+6]
        mov   cx,[bp+4]
        mov   ah,2
        jcxz  endloop
printloop:
        mov   dl,[bx]
        int   21h
        inc   bx
        loop  printloop
endloop:
        pop   dx
        pop   cx
        pop   bx
        pop   ax
        pop   bp
        ret   4
dispstr endp
```



แสต็กที่เวลานี้

? *ทำไมเราจึง PUSH BP ก่อนที่จะเก็บค่าของรีจิสเตอร์ต่าง ๆ*

ตัวอย่าง

EX เขียนโปรแกรมย่อยคำนวณค่า CHECKSUM ที่ได้จากการนำข้อมูลทั้งหมดมา XOR กัน โดยรับพารามิเตอร์ที่จะถูก PUSH ลงมาในแสต็กเรียงตามลำดับกันดังนี้ :

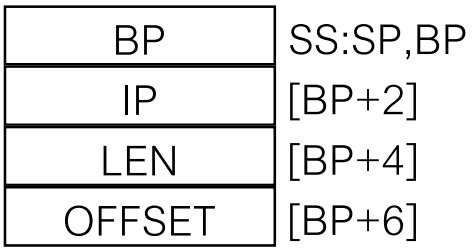
พารามิเตอร์ตัวแรกคือ ออฟเซ็ตเริ่มต้นของข้อมูล

พารามิเตอร์ตัวที่สองคือ จำนวนข้อมูล

ข้อมูลมีขนาดค่าละ 1 เวิร์ด. พารามิเตอร์ทั้งสองมีขนาด 1 เวิร์ดทั้งคู่. ให้โปรแกรมย่อยคืนค่า CHECKSUM ผ่านทางรีจิสเตอร์ AX.

➤

```
calchecksum   proc  near
        push  bp
        mov   bp,sp
        push  bx
        push  cx
        push  dx
        mov   bx,[bp+6]
        mov   cx,[bp+4]
        mov   ax,0
        jcxz  endloop
xorloop:
        xor   ax,[bx]
        inc   bx
        inc   bx
        loop  xorloop
endloop:
        pop   dx
        pop   cx
        pop   bx
        pop   bp
        ret   4
calchecksum   endp
```

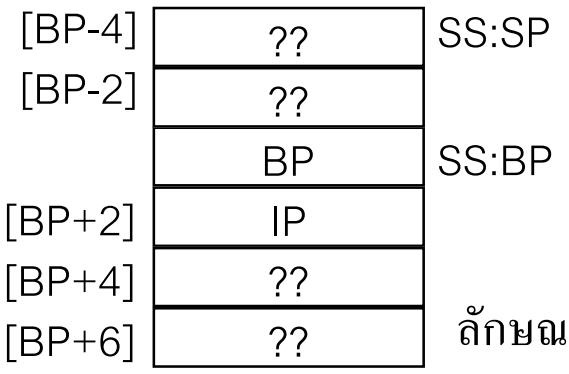


ตัวแปรภายใน

- เราสามารถใช้แสต็กสำหรับการเก็บข้อมูลชั่วคราวในโปรแกรมย่อยได้. โดยเราจะต้องจัดการกันเนื้อที่สำหรับเก็บข้อมูลเอง.

```
testproc  proc near
          push  bp
          mov   bp,sp
          sub   sp,4
```

กันเนื้อที่ไว้ 4 ไบต์



ลักษณะของแสต็ก

EX เขียนโปรแกรมย่อยหาค่าของข้อมูลที่มีความถี่มากที่สุดจากกลุ่มของข้อมูลที่ผู้ใช้ระบุมาให้. ข้อมูลแต่ละตัวจะมีค่าระหว่าง 1 ถึง 20 และมีขนาด 1 ไบต์. ผู้ใช้จะ PUSH พารามิเตอร์ให้ตามลำดับดังนี้ : ออฟเซ็ตเริ่มต้นของกลุ่มข้อมูล และ จำนวนข้อมูล. (มีขนาด 1 เวิร์ดทั้งคู่) โปรแกรมจะคืนค่าของข้อมูลที่มีความถี่มากที่สุดในรีจิสเตอร์ AL. ความถี่สูงสุดของข้อมูลจะไม่เกิน 255.

ตัวอย่าง

- ขั้นตอน
  - โปรแกรมจะต้องนับความถี่ของข้อมูลค่าต่าง ๆ
  - หาข้อมูลที่มีความถี่มากที่สุด
- จะนับความถี่อย่างไร?
  - ข้อมูลมีขอบเขตระหว่าง 1-20
  - เก็บความถี่ของข้อมูลในตารางในหน่วยความจำ
  - โปรแกรมย่อย -> ประกาศใช้ข้อมูลในแสต็ก
  - ใช้ข้อมูลขนาด 20 ไบต์

```
findmost  proc  near
          push  bp
          mov   bp,sp
          sub   sp,20

            ...

          add   sp,20
          pop   bp
          ret   4
findmost  endp
```