โปรแกรมย่อยขั้นต้น

- การประกาศ

```
procname   PROC  NEAR
           …     ;program
           RET
procname   ENDP
```

- การเรียกใช้

```
CALL   procname
```

```
;print Hex digit
;input al<-digit
;affect : dl,ah
printhexdigit proc  near
              mov   ah,2
              mov   dl,al
              add   dl,'0'
              cmp   al,10
              jb    printit
              add   dl,'A'-'0'-10
printit:      int   21h
              ret
printhexdigit endp
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

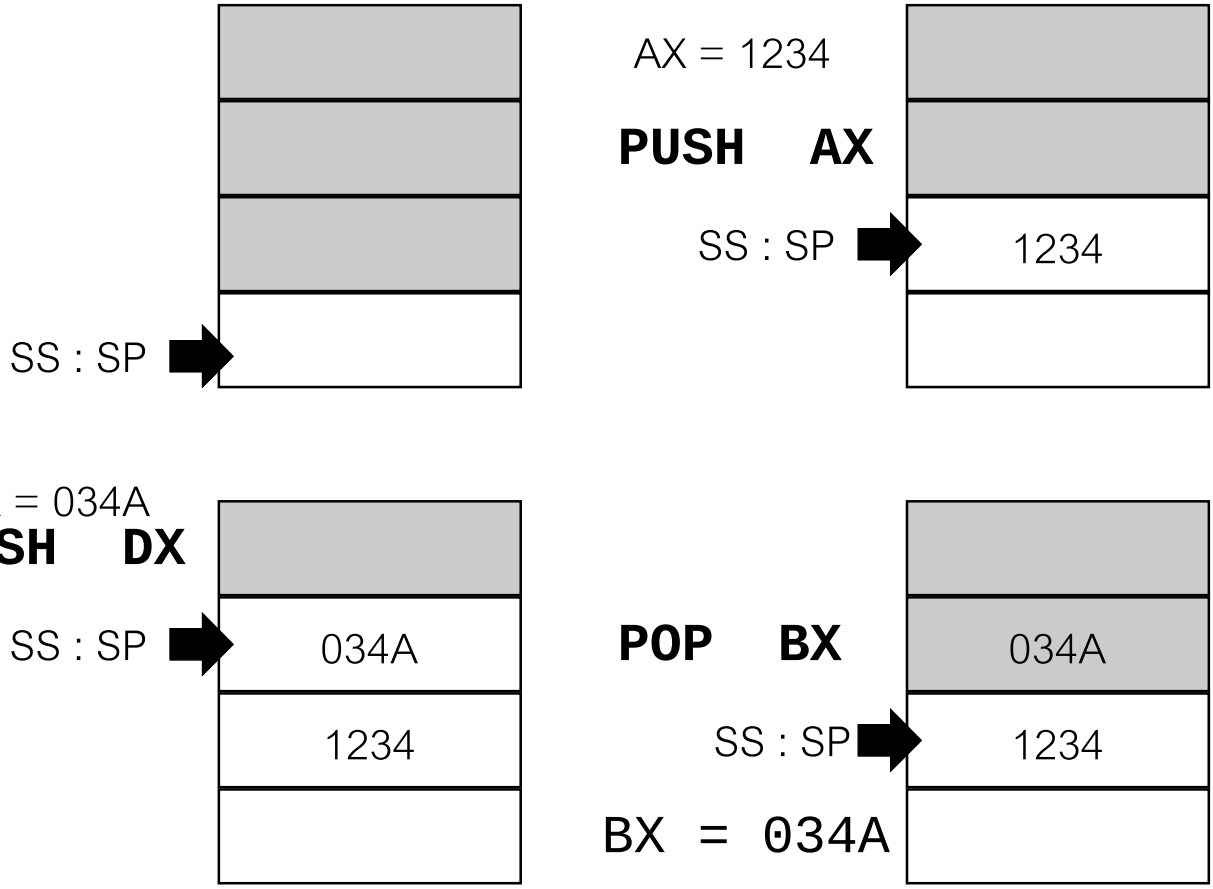
โปรแกรมย่อยขั้นต้น

```
; print a 2-digit hex number
; the digits is stored in bh,bl
                ...
                mov     al,bh
                call    printhexdigit
                mov     al,bl
                call    printhexdigit
                ...
```

- **ผลกระทบข้างเคียง**
  - การทำงานของโปรแกรมย่อยอาจมีผลกระทบข้างเคียงกับรีจิสเตอร์อื่น ๆ ได้
  - เราควรป้องกันการเกิดผลกระทบนี้
  - เก็บรักษาค่าเดิมของรีจิสเตอร์ -> ***ใช้ stack***
- **คำสั่งเกี่ยวกับ stack**
  - PUSH *reg16* : ใส่ค่าลงใน stack
  - POP *reg16* : อ่านค่าขึ้นมาจาก stack

การทำงานของคำสั่ง PUSH และ POP

- โดยปกติรีจิสเตอร์ SS:SP จะเก็บตำแหน่งบนสุดของแสต็ก. เมื่อเราสั่ง PUSH หรือ POP กับแสต็ก หน่วยประมวลผลจะปรับค่าของรีจิสเตอร์ SP ให้มีค่าชี้ตำแหน่งบนสุดของแสต็กอีกครั้ง.



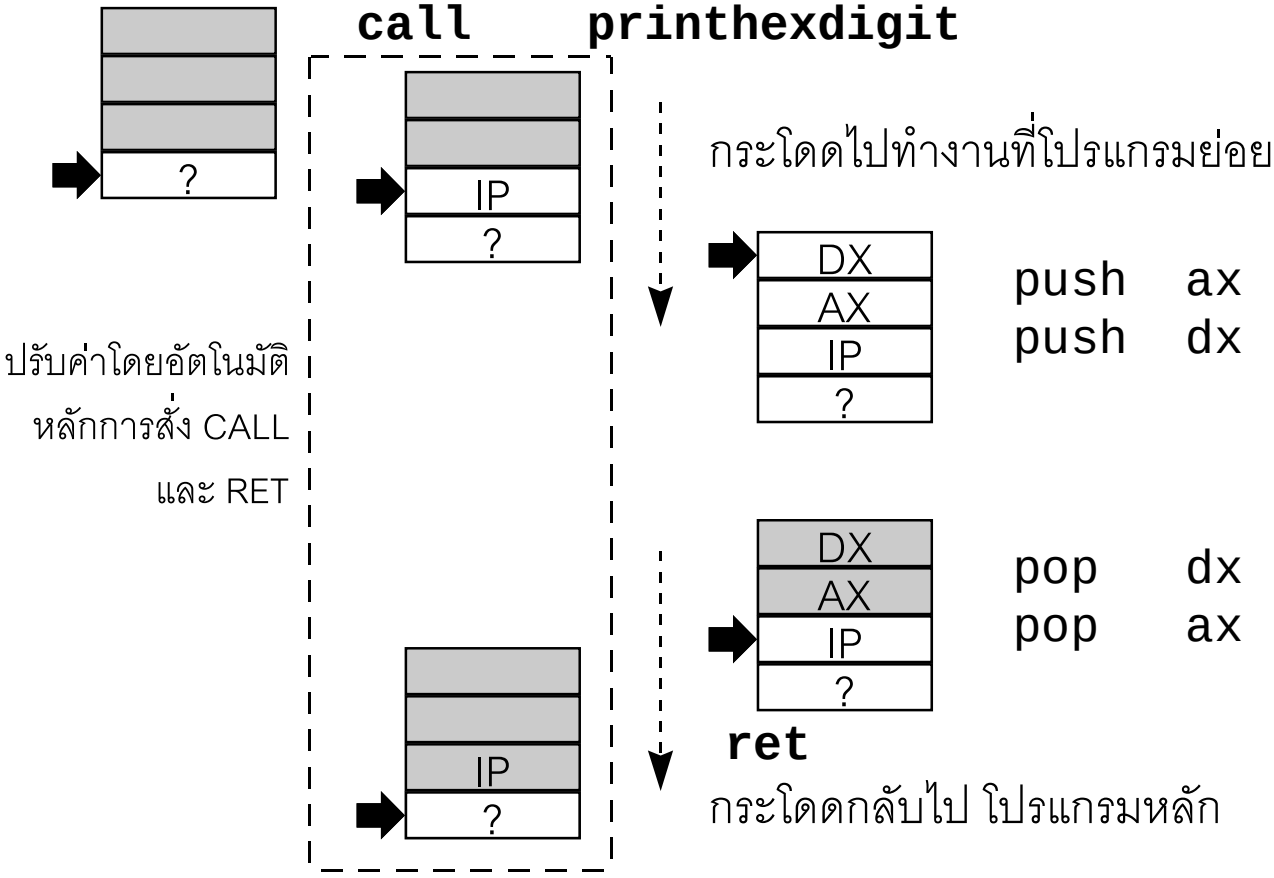
โปรแกรมย่อยขั้นต้น

```
;print Hex digit
;input al<-digit
printhexdigit proc  near
              push  ax
              push  dx
              mov   ah,2
              mov   dl,al
              add   dl,'0'
              cmp   al,10
              jb    printit
              add   dl,'A'-'0'-10
printit:      int   21h
              pop   dx
              pop   ax
              ret
printhexdigit endp
```


สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

โปรแกรมย่อยขั้นต้น

- **การเปลี่ยนแปลงของ stack เมื่อมีการเรียกโปรแกรมย่อย**



- การผ่านค่า parameter ให้กับโปรแกรมย่อยจากตัวอย่างใช้การผ่านค่าผ่านทางรีจิสเตอร์. ในบทถัด ๆ ไปเราจะได้เรียนรู้การผ่านค่าผ่านทาง stack ซึ่งเป็นแบบนิยมใช้ในภาษาระดับสูง.