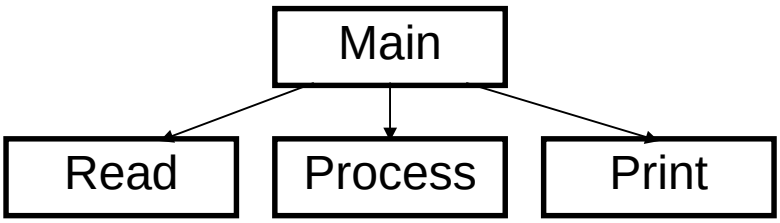
Lecture 10

Internal Subprograms

Text:
(4th Edition: Chapter 8)
(5th Edition: Chapter 7)

Calling Procedures

Breaking down large problems into smaller tasks is a well-established programming practice.



The main program in assembly language is a procedure, and its starting point is indicated by the operand of the END directive.

```
BEGIN   PROC   FAR  ; the main prog.
          .
          .
          .
BEGIN   ENDP
          .
          END   BEGIN
```

Two instructions are used for procedures:

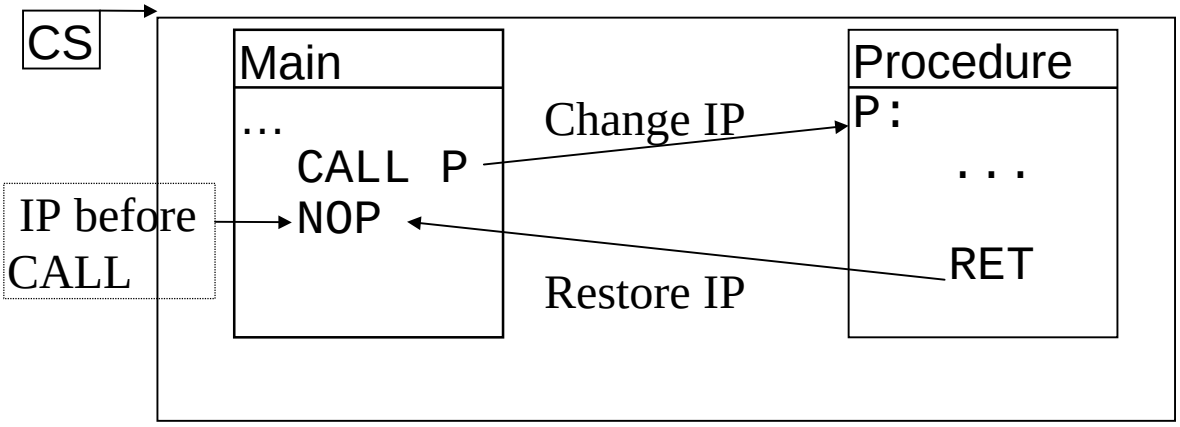
CALL To call the procedure

```
[label:]    CALL  <procedure>
```

RET To return from the procedure

```
[label:]   RET    <immediate>
```

Issues in CALLING a procedure



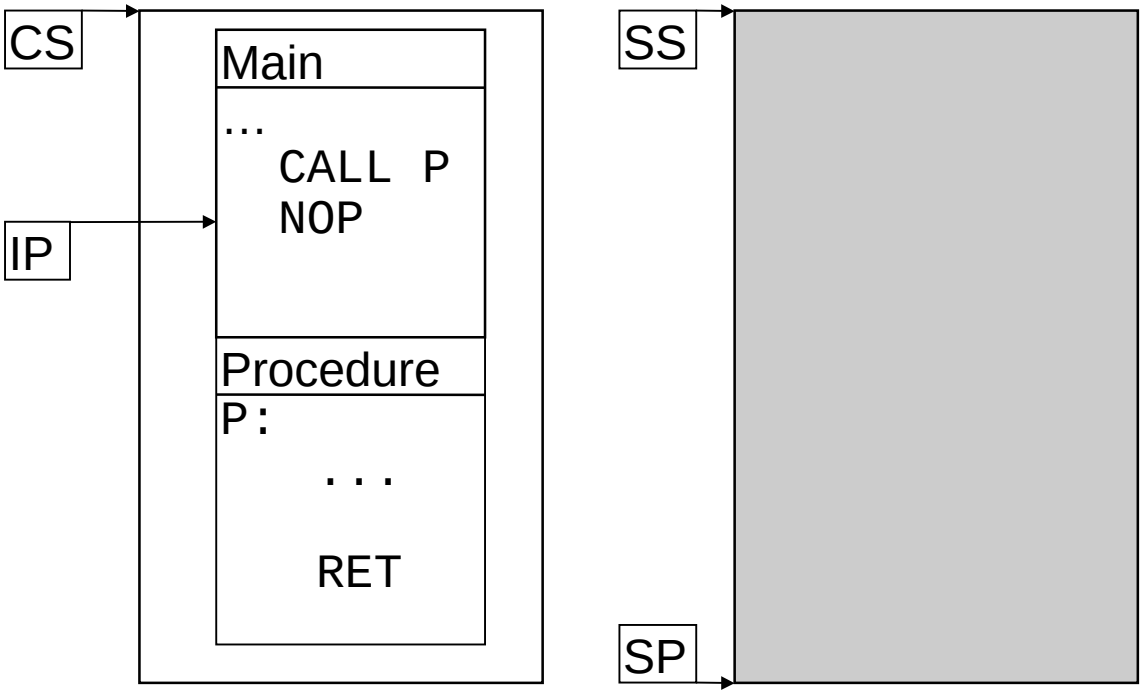
The call statement changes the IP so that it contains the offset in the Code Segment for the Procedure.

Before the change, the IP pointed to the next instruction in the main program. This value must be saved before the IP is changed.

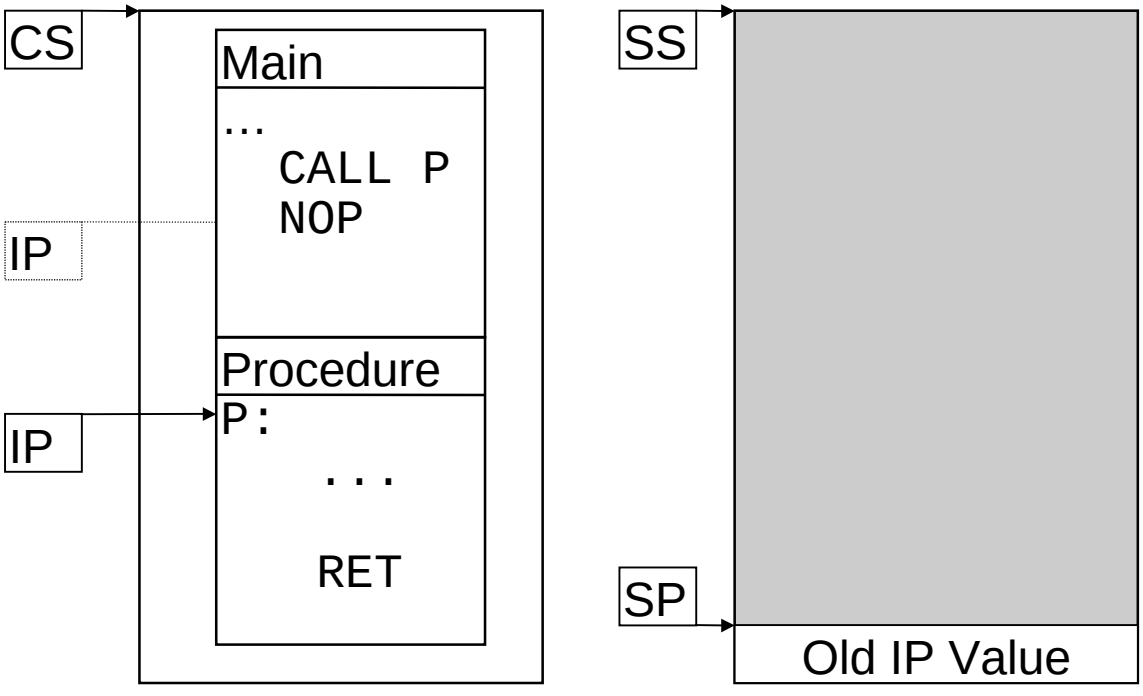
The RETurn statement restores the IP with the saved value of the main program instruction which followed the CALL.

The CALL statement saves the return point IP in the program's STACK.

Before CALL:



After CALL:



The previous example was a **NEAR** procedure call because the procedure was in the same segment as the calling program.

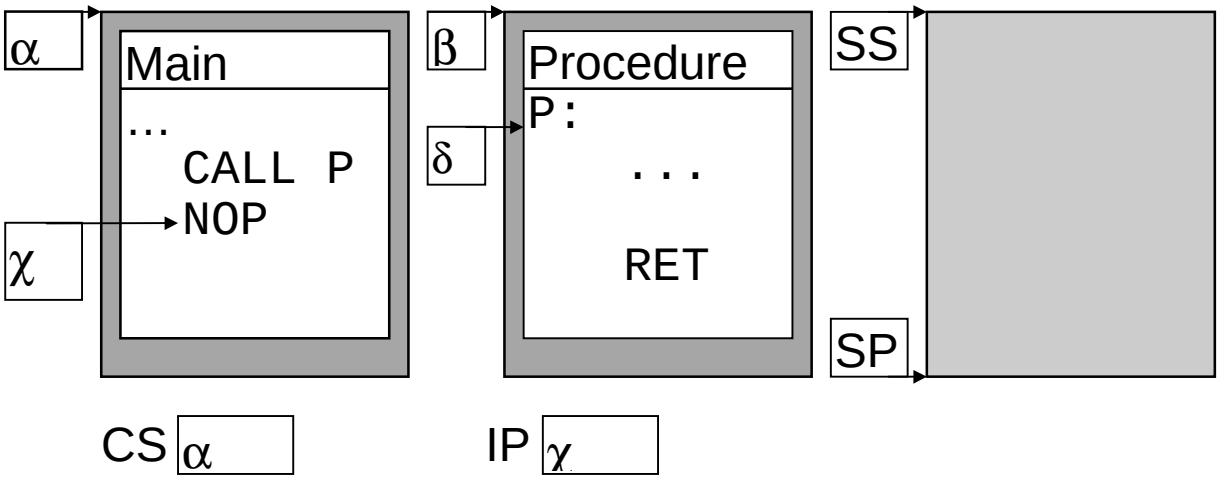
Example:

```
        page 60,132
TITLE   P08CALLP (EXE)  Calling procedures
        .MODEL  SMALL
        .STACK  64
        .DATA
;---------------------------------------
        .CODE
BEGIN   PROC    FAR
        CALL    B10         ;Call B10
;       ...
        MOV     AX,4C00H   ;Exit to DOS
        INT     21H
BEGIN   ENDP
;---------------------------------------
B10     PROC    NEAR
        CALL    C10         ;Call C10
;       ...
        RET                 ;Return to
B10     ENDP               ;  caller
;---------------------------------------
C10     PROC    NEAR
;       ...
        RET                 ;Return to
C10     ENDP                    ;  caller
;---------------------------------------
        END     BEGIN
```

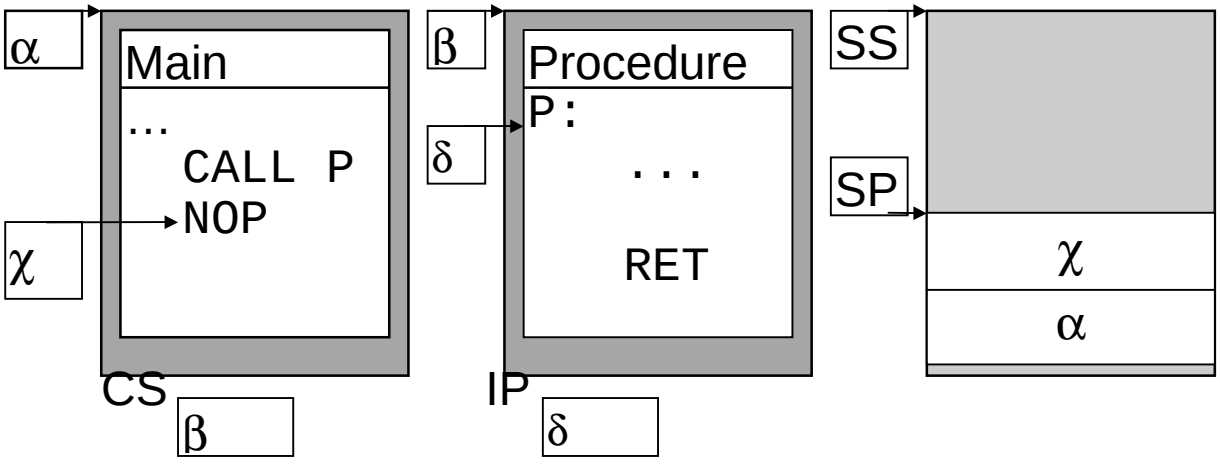
FAR Procedures

- The procedure is in a different code segment.
- There are TWO Code Segment addresses
- The address of the CURRENT (calling) segment is in the CS register.
- The CALL must save BOTH the IP and the CS register values.
- The RETurn must restore both.

Before Call:



After Call:



Exercises - Lecture 10

1. Fill in the main program so that it calls the subprogram SUMODD, and then write the subprogram SUMODD as a near procedure. The variable THESUM should contain the sum of the odd numbers from 1 to LASTODD.

```
DATASG    SEGMENT PARA
LASTODD   DW      21
THESUM            DW    ?
DATASG    ENDS
;
CODESG    SEGMENT PARA
;
BEGIN     PROC    FAR   ; THE MAIN PROGRAM BEGINS HERE
;
          MOV     AX,DATASG
          MOX     DS,AX

          _______________ ; SUM THE ODD NUMBERS
          MOV     AX,4C00H
          INT     21H
BEGIN     ENDP
;
;  SUM THE ODD NUMBERS FROM 1 TO 'LASTODD'
;
```

_____

_____

_____

_____

_____

_____

_____

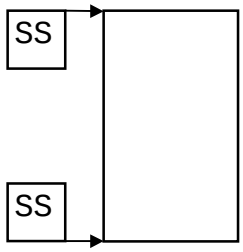
_____

_____

_____

2. Examine the following registers and determine what their contents will be if the next instruction is a near call to CS:0092. Write any changes to the stack.

```
CS: 4593        New:_______
IP: 0032        New:_______
SS: 5328        New:_______
SP: 0004        New:_______
DS: 4655        New:_______
```



3. Examine the following registers and determine what their contents will be if the next instruction is a far call to 4866:0092. Write any changes to the stack.

```
CS: 4593        New:_______
IP: 0032        New:_______
SS: 5328        New:_______
SP: 0004        New:_______
DS: 4655        New:_______
```