III. STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING TECHNIQUES

Modular Programming

More Examples

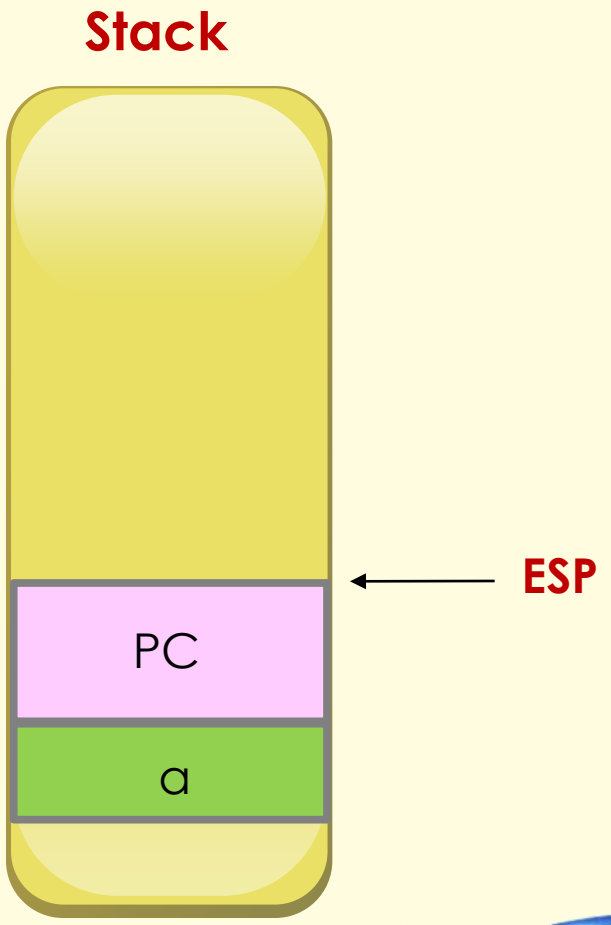
```
int abc (int n) {
    int result=n;
    while(n>1) {
        n--;
        result=result*n;
    }
    return result;
}

r = abc (a);
```

```
;subprogram call

sub esp, 2
push word [a]
call abc
pop word [r]
```

Stack

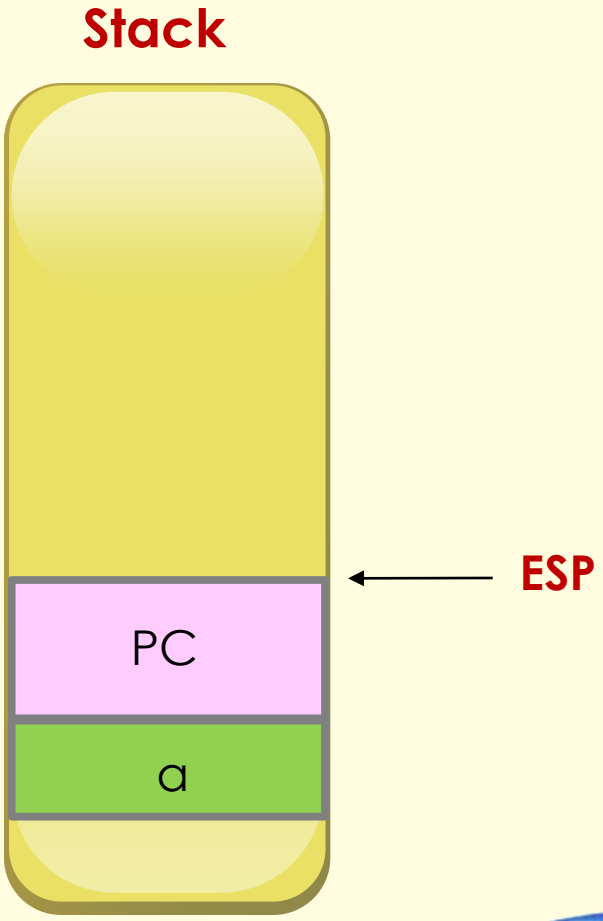
subprogram call:

sub esp, 2
push word [a]
call abc
pop word [r]



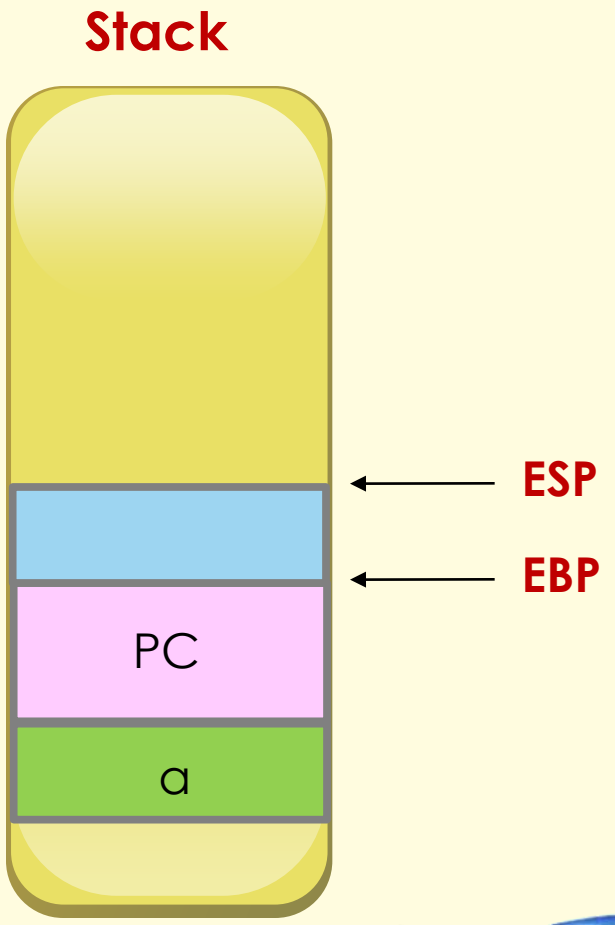
More Examples

```
int abc (int n) {
    int result=n;
    while(n>1) {
        n--;
        result=result*n;
    }
    return result;
}
```



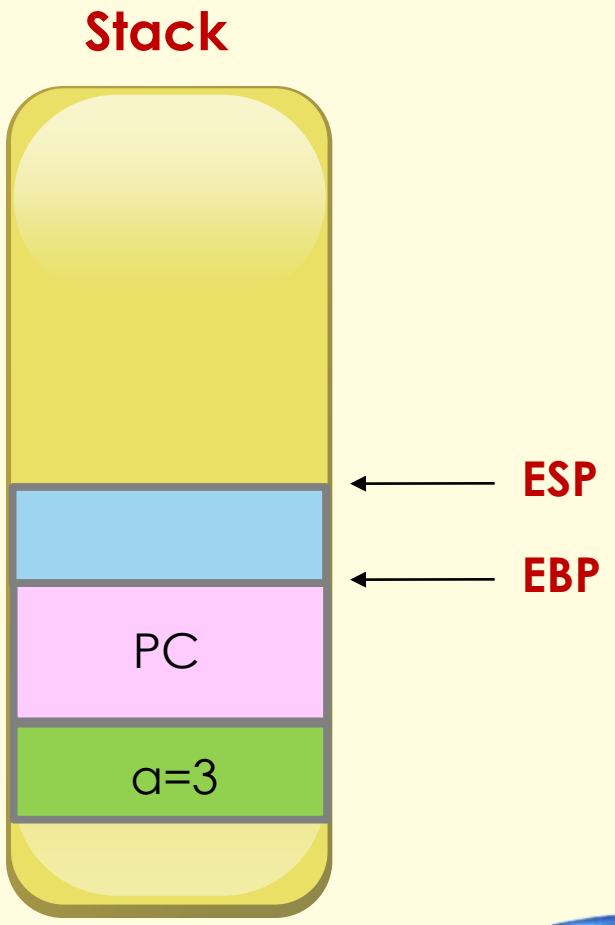
More Examples

```
int abc (int n) {
    int result=n;
    while(n>1) {
        n--;
        result=result*n;
    }
    return result;
}

;subprogram
  abc:
    mov ebp, esp
    sub esp, 2
```



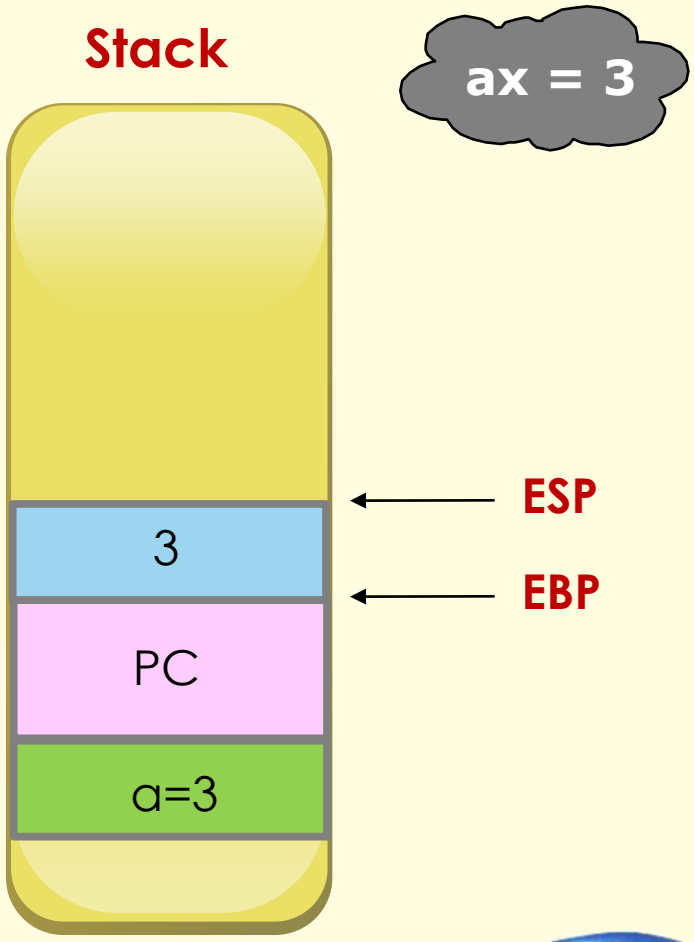
More Examples

```
int abc (int n) {
    int result=n;
    while(n>1) {
        n--;
        result=result*n;
    }
    return result;
}

;subprogram
  abc:
    mov ebp, esp
    sub esp, 2
```



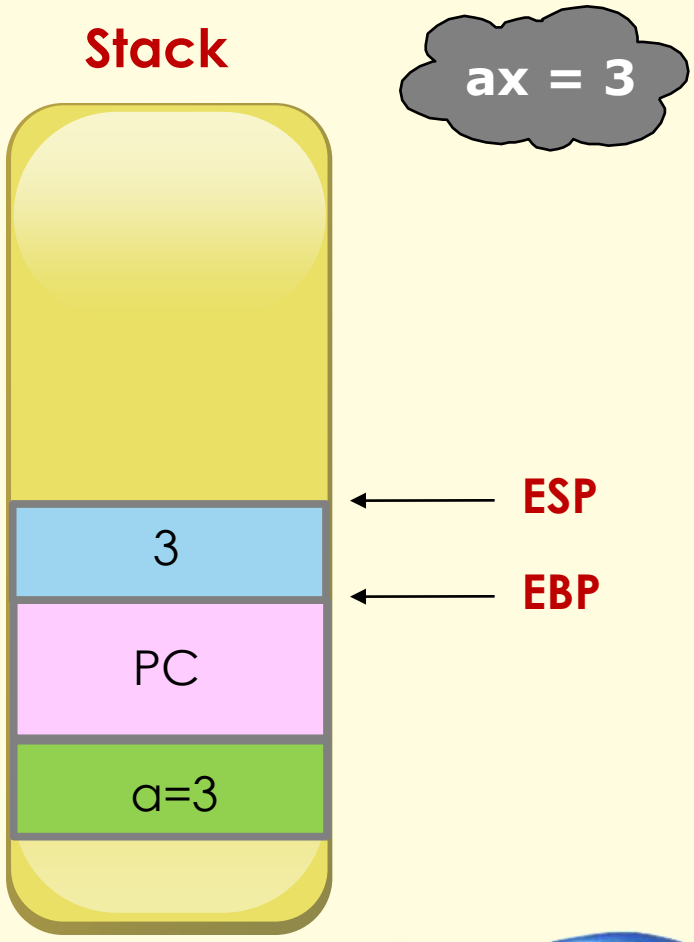
More Examples

```
int abc (int n) {
     int result=n;
     . . .
     return result;
}

;subprogram
   abc:
     mov ebp, esp
     sub esp, 2
     mov ax, [ebp+4]
     mov word[ebp-2], ax
```



More Examples

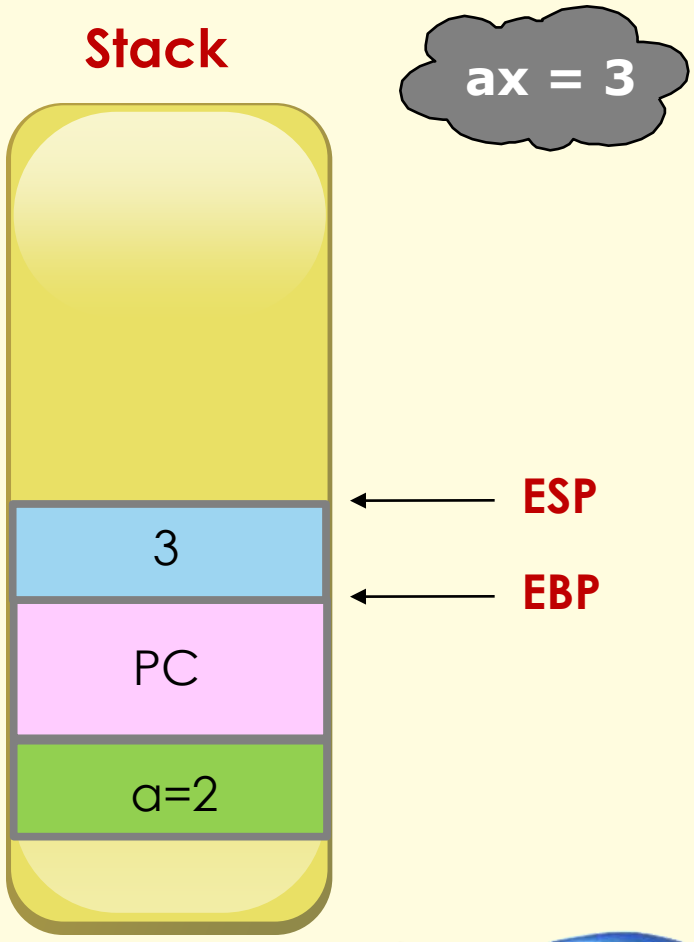
```
int abc (int n) {
    int result=n;
    while(n>1) {
        n--;
        result=result*n;
    }
    return result;
}
```



More Examples

```
. . .
while(n>1) {
        n--;
        result=result*n;
}
. . .
```

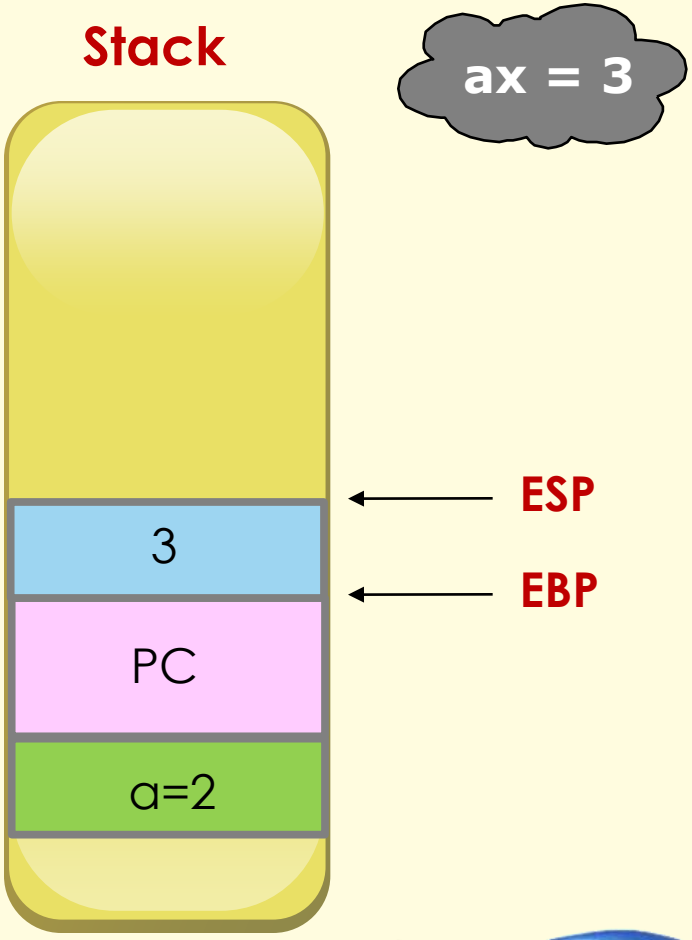
```
;subprogram
while:
        cmp word[ebp+4], 1
        jng exit
        dec word[ebp+4]
```



More Examples

```
. . .
        result=result*n;
. . .
```

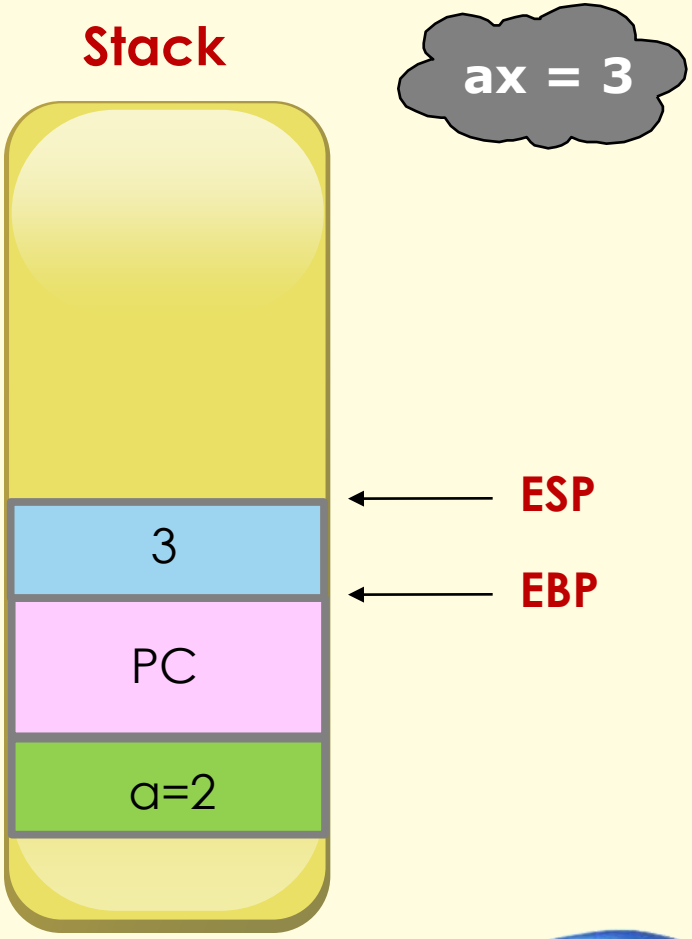
```
;subprogram
while:
      cmp word[ebp+4], 1
      jng exit
      dec word[ebp+4]
      mov ax, [ebp-2]
```



More Examples

```
. . .
        result=result*n;
. . .
```

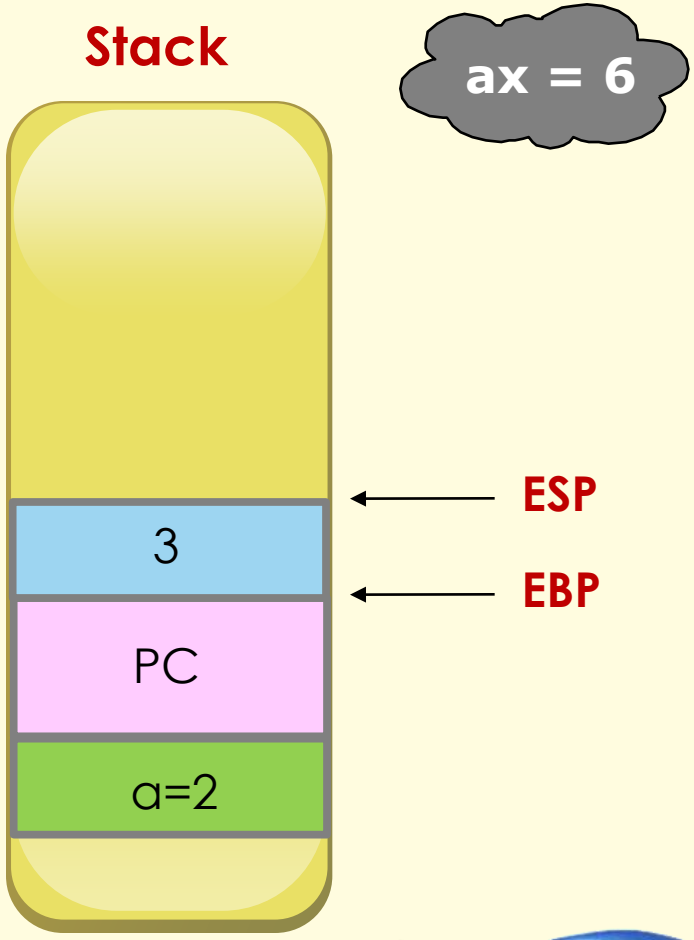
```
;subprogram
while:
      cmp word[ebp+4], 1
      jng exit
      dec word[ebp+4]
      mov ax, [ebp-2]
      mul word[ebp+4]
```



More Examples

```
. . .
        result=result*n;
. . .
```

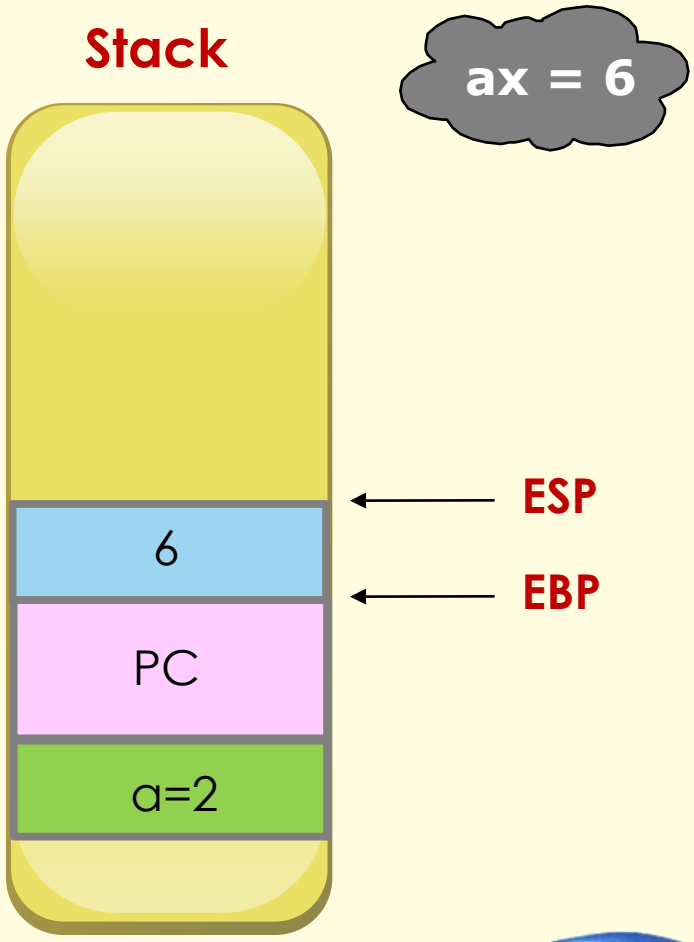
```
;subprogram
while:
      cmp word[ebp+4], 1
      jng exit
      dec word[ebp+4]
      mov ax, [ebp-2]
      mul word[ebp+4]
```



More Examples

```
. . .
        result=result*n;
. . .
```

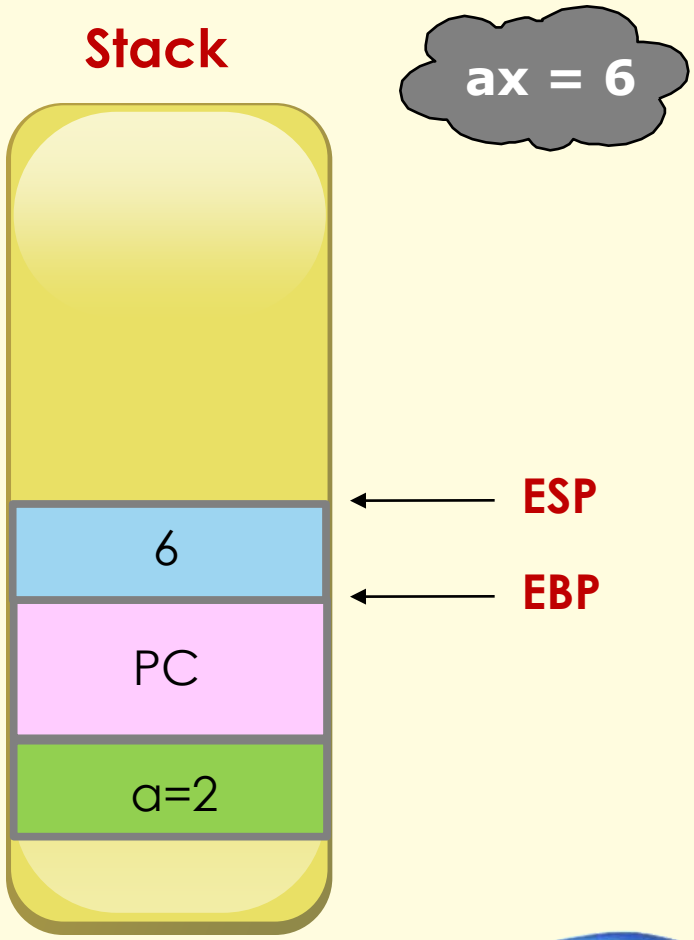
```
;subprogram
while:
        cmp word[ebp+4], 1
        jng exit
        dec word[ebp+4]
        mov ax, [ebp-2]
        mul word[ebp+4]
        mov word[ebp-2], ax
```



More Examples

```
. . .
        result=result*n;
. . .
```

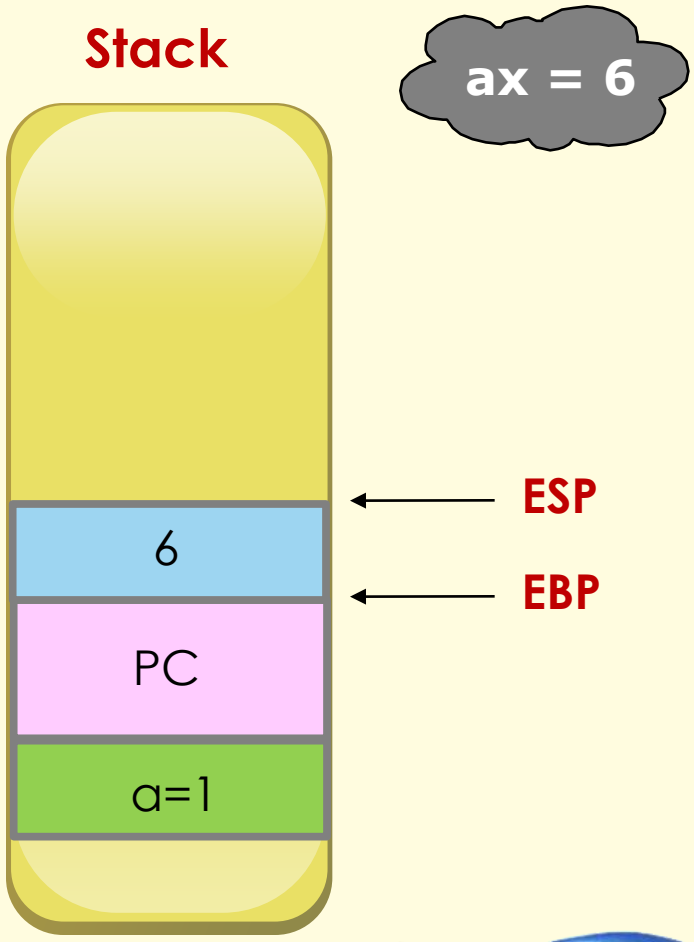
```
;subprogram
while:
        cmp word[ebp+4], 1
        jng exit
        dec word[ebp+4]
        mov ax, [ebp-2]
        mul word[ebp+4]
        mov word[ebp-2], ax
        jmp while
```



More Examples

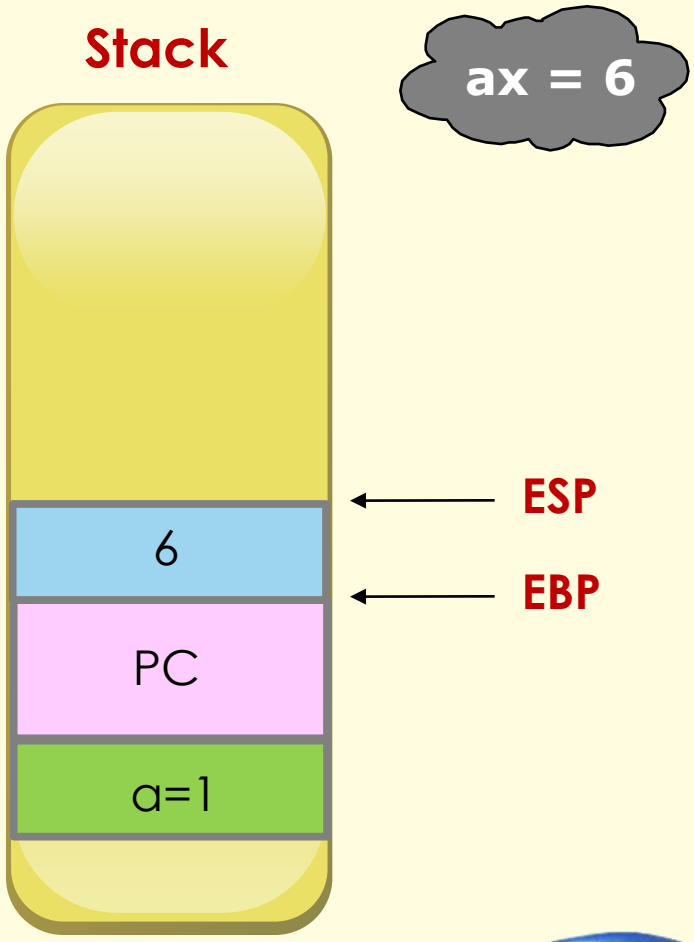
```
. . .
        result=result*n;
. . .
```

```
;subprogram
while:
        cmp word[ebp+4], 1
        jng exit
        dec word[ebp+4]
        mov ax, [ebp-2]
        mul word[ebp+4]
        mov word[ebp-2], ax
        jmp while
```



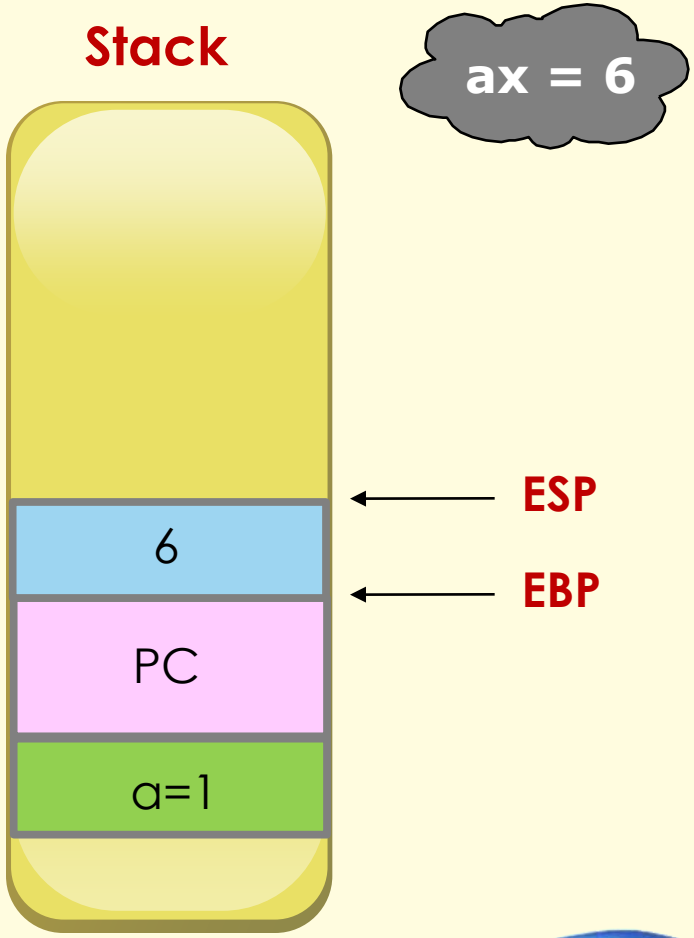
More Examples

```
int abc (int n) {
    int result=n;
    while(n>1) {
        n--;
        result=result*n;
    }
    return result;
}
```



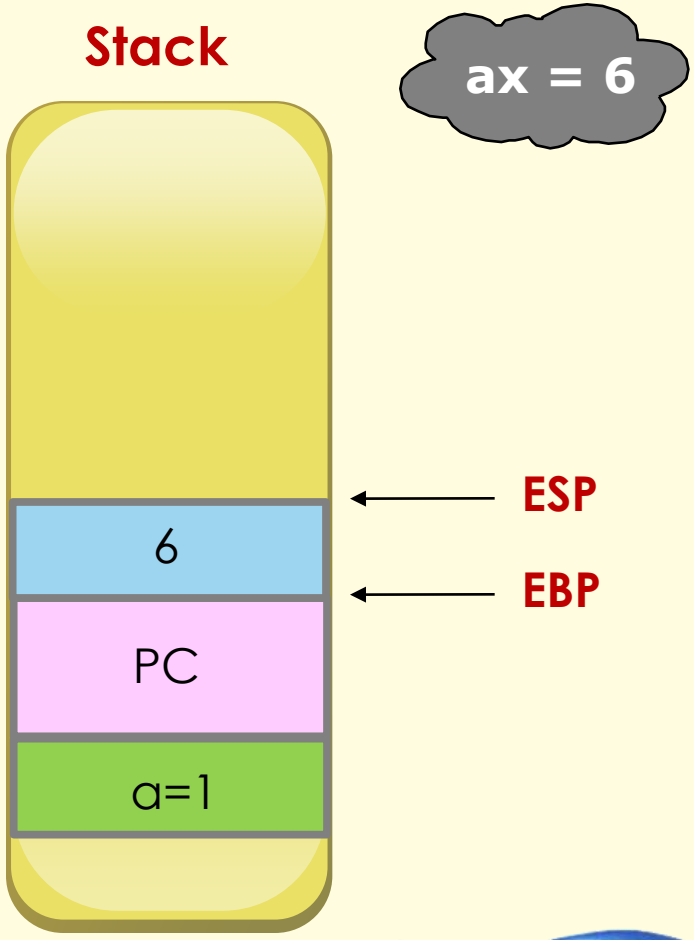
More Examples

```
;subprogram
abc:
    ...
    mov word[ebp-2], ax
    while:
     cmp word[ebp+4], 1
     jng exit
     dec word[ebp+4]
     mov ax, [ebp-2]
     mul word[ebp+4]
     mov word[ebp-2], ax
     jmp while
```



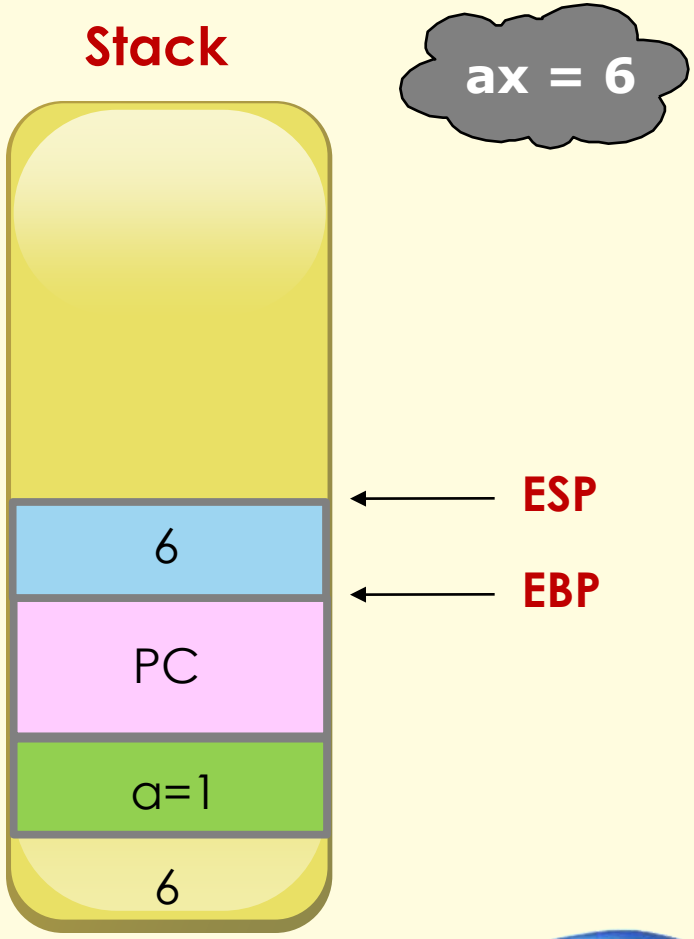
More Examples

```
;subprogram
abc:
    ...
    mov word[ebp-2], ax
    while:
     . . .
     jmp while
    exit:
```



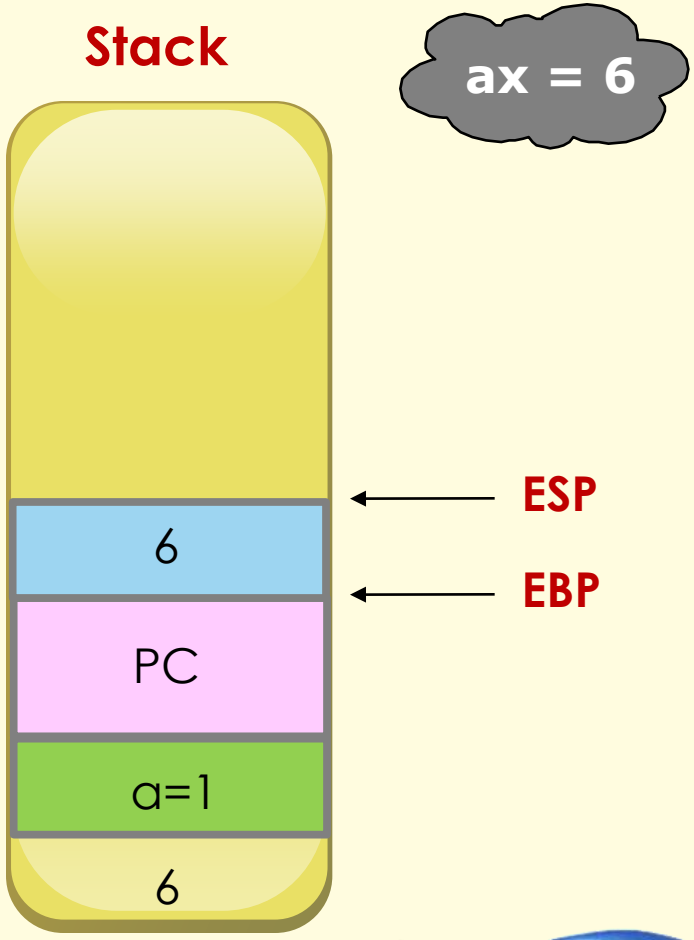
More Examples

```
;subprogram
abc:
    ...
    mov word[ebp-2], ax
    while:
     . . .
     jmp while
    exit:
```



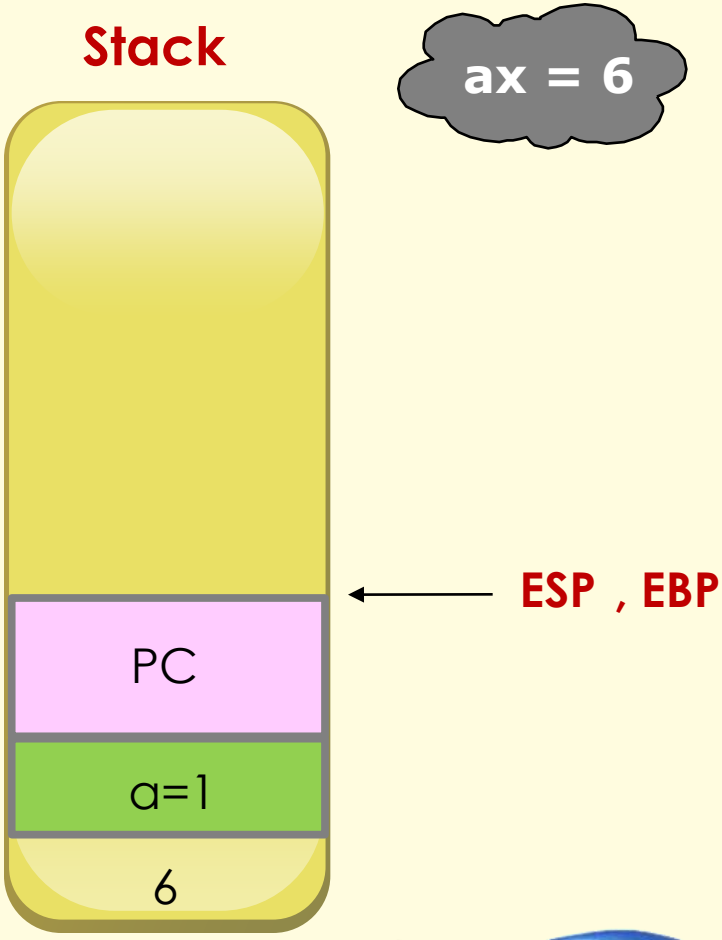
More Examples

```
;subprogram
abc:
    ...
    mov word[ebp-2], ax
    while:
     . . .
     jmp while
    exit:
     mov ax, [ebp-2]
     mov word[ebp+6], ax
```



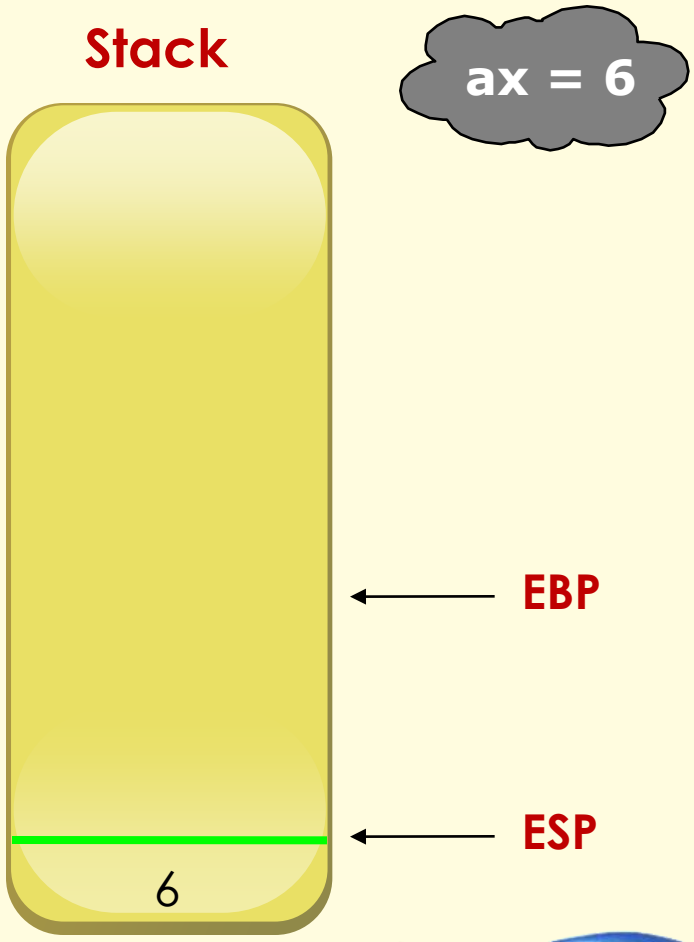
More Examples

```
;subprogram
abc:
    ...
    mov word[ebp-2], ax
    while:
     . . .
     jmp while
    exit:
     mov ax, [ebp-2]
     mov word[ebp+6], ax
     add esp, 2
```



More Examples

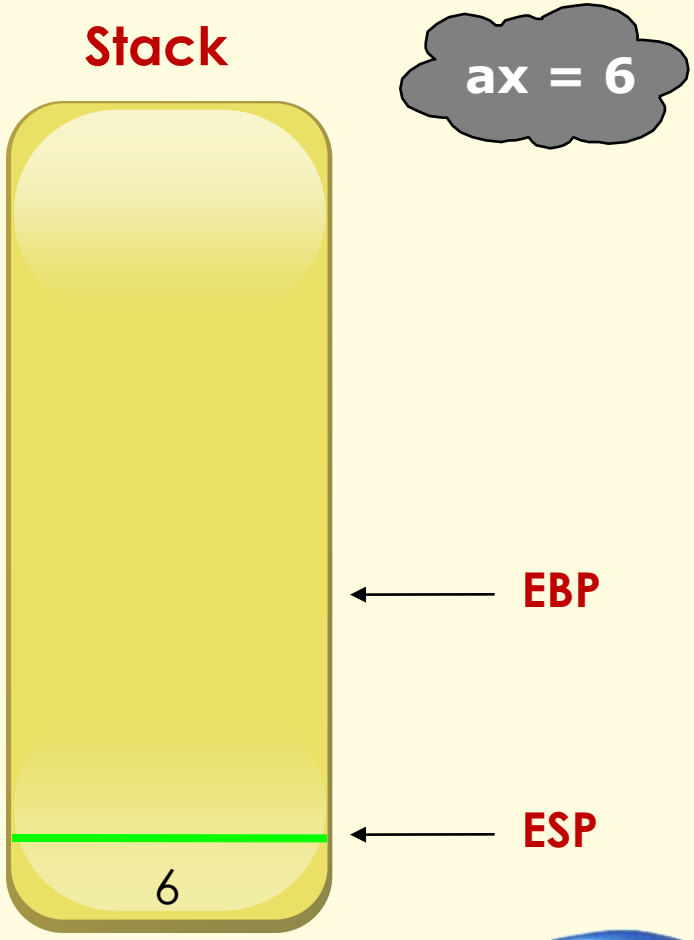
```
;subprogram
abc:
    ...
    mov word[ebp-2], ax
    while:
     . . .
     jmp while
    exit:
     mov ax, [ebp-2]
     mov word[ebp+6], ax
     add esp, 2
     ret 2
```



More Examples

;subprogram call

sub sp, 2
push word [a]
call abc
pop word [r]



Pointers Review (C Programming)

- Pointers
  - variables which hold the address of other variables
  - tell user where a variable resides in memory
  - can access a variable indirectly

```
int *p;          int x;
x = 10;
p = &x;
*p = 100;
```

Variables and their Addresses

- Variables are just locations in memory.
- Variable name == Human readable location of variable in memory.
- Variable location in memory
  - Offset from the start of the Data Segment
- num
  - address of variable
- [num]
  - value at memory address DS+num

Machine Equivalent of Pointers

```
int *EBX;
int x, y;

x = 100;
EBX = &x;
y = *EBX – 90
```

Machine Equivalent of Pointers

```
int *EBX;
int x, y;

x = 100;
EBX = &x;
y = *EBX – 90
```

```
x  dw  0
y dw 0

mov word [x], 100
mov ebx, x
mov ax, [ebx]
sub ax, 90
mov word[y], ax
```

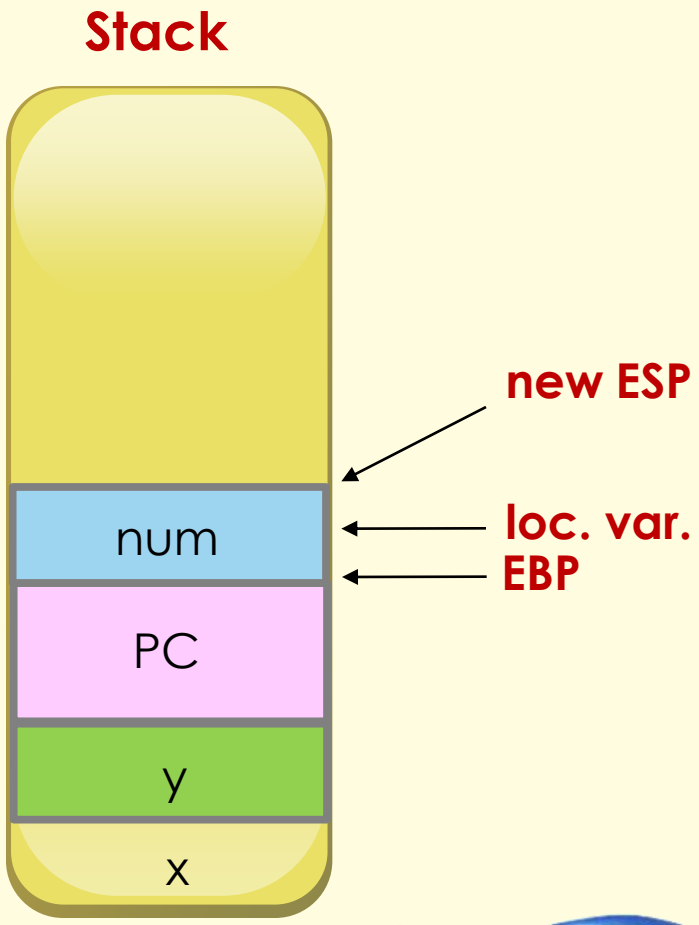
Caution

- Registers and globally declared variables may be changed within any subprogram.

- Whatever the final values of registers and globally declared variables are at the end of the subprogram will be the value they hold when they return to the calling subprogram.

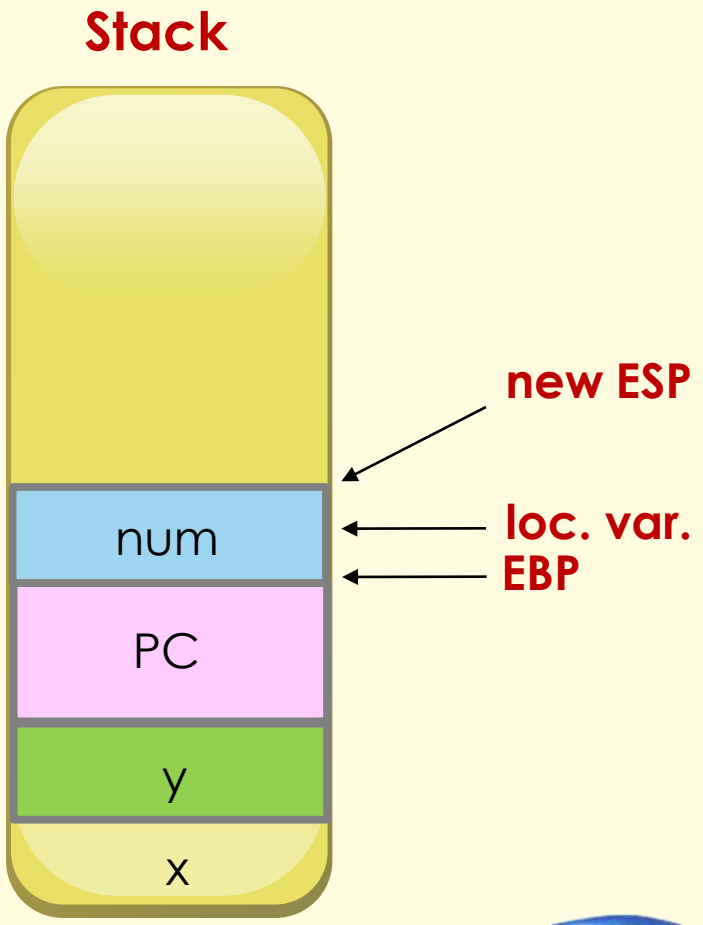
Recall: Stack

```
sum:
    mov  ebp, esp
    sub   esp, 2
    mov  ax, [ebp + 6]
    add   ax, [ebp + 4]
    mov  [ebp – 2], ax
    add   esp, 2
    ret    4
```



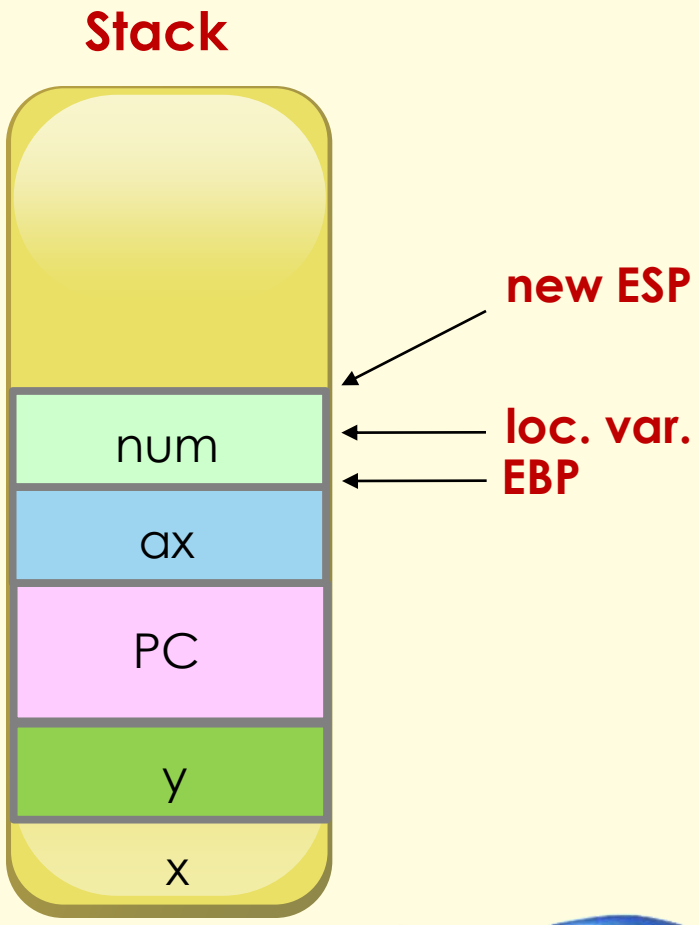
Recall: Stack

```
sum:
    push ax
    mov  ebp, esp
    sub  esp, 2
    mov  ax, [ebp + 6]
    add  ax, [ebp + 4]
    mov  [ebp – 2], ax
    add  esp, 2
    ret  4
```



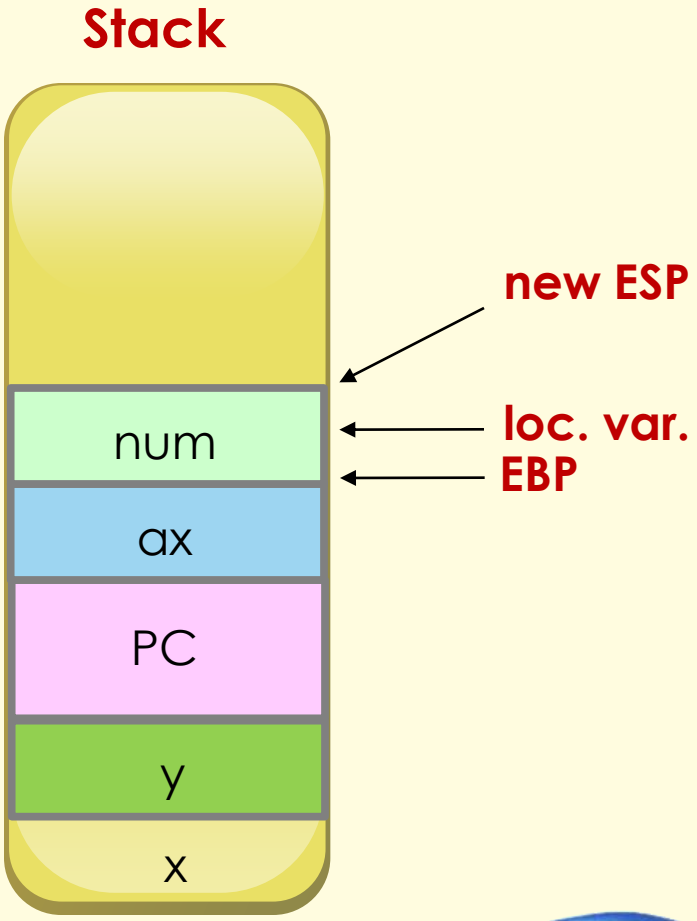
Recall: Stack

```
sum:
    push ax
    mov  ebp, esp
    sub  esp, 2
    mov  ax, [ebp + 8]
    add  ax, [ebp + 6]
    mov  [ebp – 2], ax
    add  esp, 2
    ret   4
```



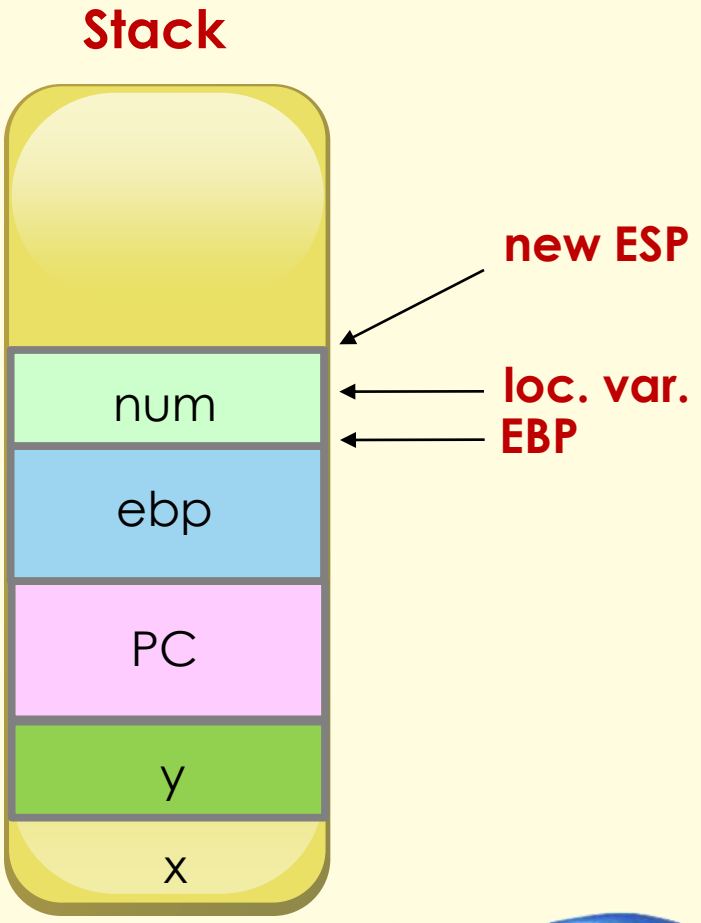
Recall: Stack

```
sum:
    push ax
    mov  ebp, esp
    sub   esp, 2
    mov  ax, [ebp + 8]
    add   ax, [ebp + 6]
    mov  [ebp – 2], ax
    add   esp, 2
    pop ax
    ret    4
```



Recall: Stack

```
sum:
    push ebp
    mov  ebp, esp
    sub   esp, 2
    mov  ax, [ebp + 10]
    add   ax, [ebp + 8]
    mov  [ebp – 2], ax
    add   esp, 2
    pop ebp
    ret    4
```



Saving Registers

- At the start of the subprogram, save all registers not just EBP.
- At the end of the subprogram, restore all registers not just EBP.
- **pusha**
  - EAX ECX EDX EBX ESP EBP ESI EDI
- **popa**

More Examples

```
void more                          ;subprogram call
(int *x, int y, int z){
   *x = y * z ;
}

more (&d, e, f);
```

More Examples

```
void more
(int *x, int y, int z){
    *x = y * z ;
}

more (&d, e, f);
```

```
;subprogram call

push d
push word[e]
push word[f]
call more
```

More Examples

```
void more
(int *x, int y, int z){
   *x = y * z ;
}

more (&d, e, f);
```

```
;subprogram

more:
```

More Examples

```
void more
(int *x, int y, int z){
   *x = y * z ;
}

more (&d, e, f);
```

```
;subprogram

more:
    mov ebp, esp


    ret 8
```

More Examples

```
void more
(int *x, int y, int z){
   *x = y * z;
}

more (&d, e, f);
```

```
;subprogram

more:
    mov ebp, esp
    mov ax, [ebp+6]
    mul word[ebp+4]


    ret 8
```

More Examples

```
void more
(int *x, int y, int z){
   *x = y * z;
}

more (&d, e, f);
```

```
;subprogram

more:
    mov ebp, esp
    mov ax, [ebp+6]
    mul word[ebp+4]
    mov ebx, [ebp+8]
    mov [ebx], ax
    ret 8
```

Quiz

```
quiz:
    mov ebp, esp
    mov ax, [ebp+6]
    while:
        cmp word[ebp+4], 1
        jng exit
        add ax, [ebp+6]
        dec word[ebp+4]
        jmp while
    exit:
        mov word[ebp+8], ax
        ret 4
```