Review function call in Assembly

- **Push parameters onto the stack, from right to left**
- **Call the function**
- **Save and update the %ebp**
- **Save registers used for temporaries**
- **Allocate local variables**
- **Perform the function's purpose**
- **Release local storage**
- **Restore saved registers**
- **Restore the old %ebp**
- **Return from the function**

Factorial Example

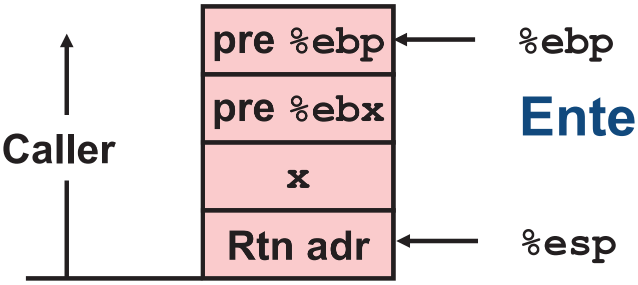
```
int rfact(int x)
{
  int rval;
  if (x <= 1)
    return 1;
  rval = rfact(x-1);
  return rval * x;
}
```

Registers

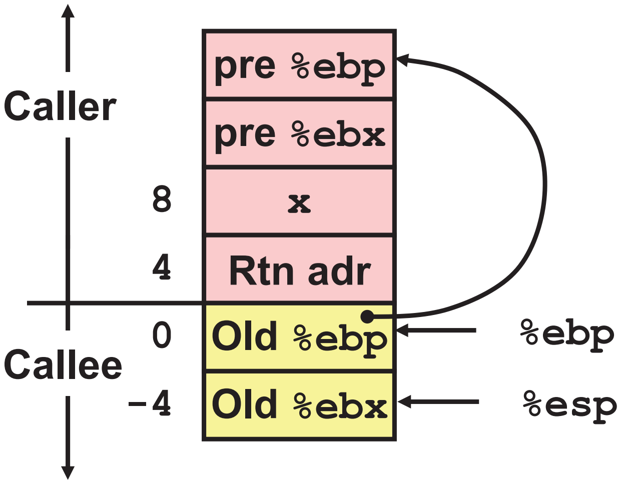
- %eax used without first saving
- %ebx used, but save at beginning & restore at end

```
rfact:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    movl 8(%ebp),%ebx
    cmpl $1,%ebx
    jle .L78
    leal -1(%ebx),%eax
    pushl %eax
    call rfact
    imull %ebx,%eax
    jmp .L79
    .align 4
.L78:
    movl $1,%eax
.L79:
    movl -4(%ebp),%ebx
    movl %ebp,%esp
    popl %ebp
    ret
```

Entering Stack



```
rfact:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
```



Recursion

```
  movl 8(%ebp),%ebx   # ebx = x
  cmpl $1,%ebx        # Compare x : 1
  jle .L78            # If <= goto Term
  leal -1(%ebx),%eax  # eax = x-1
  pushl %eax          # Push x-1
  call rfact          # rfact(x-1)
  imull %ebx,%eax     # rval * x
  jmp .L79            # Goto done
.L78:               # Term:
  movl $1,%eax        # return val = 1
.L79:               # Done:
```

```
int rfact(int x)
{
  int rval;
  if (x <= 1)
    return 1;
  rval = rfact(x-1) ;
  return rval * x;
}
```

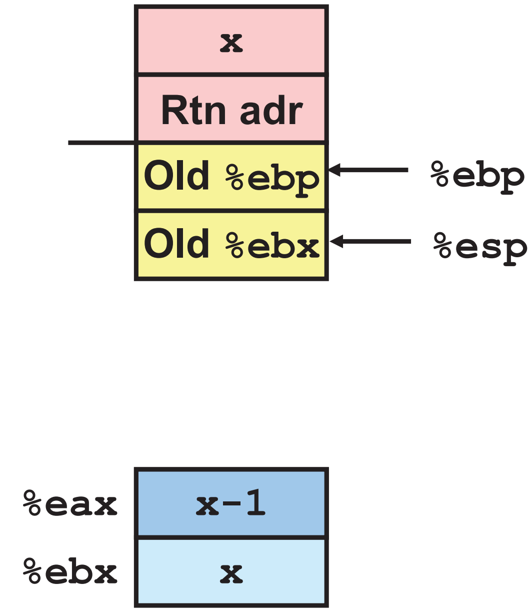
Registers

`%ebx` Stored value of x

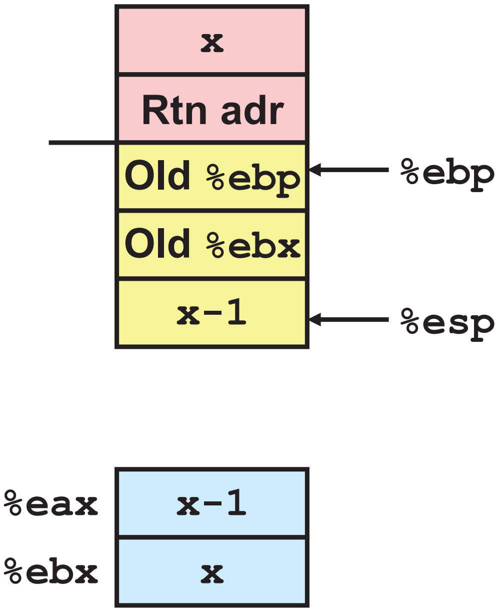
`%eax`

- Temporary value of `x-1`
- Returned value from `rfact(x-1)`
- Returned value from this call

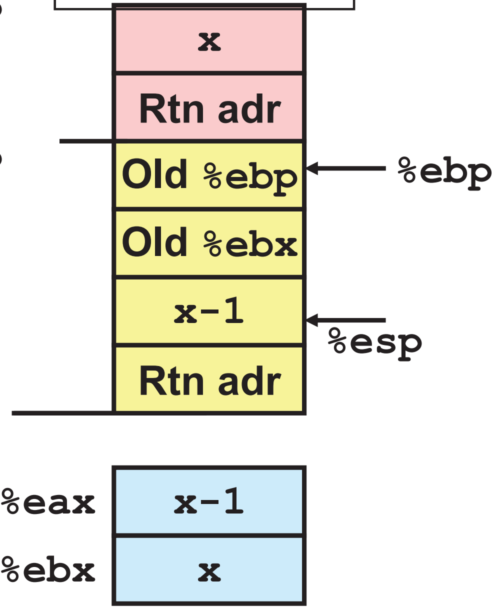
```
leal -1(%ebx),%eax
```



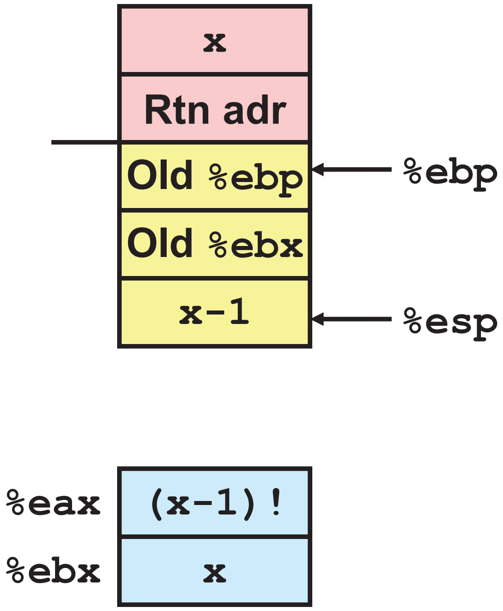
```
pushl %eax
```



```
cal rfact
```

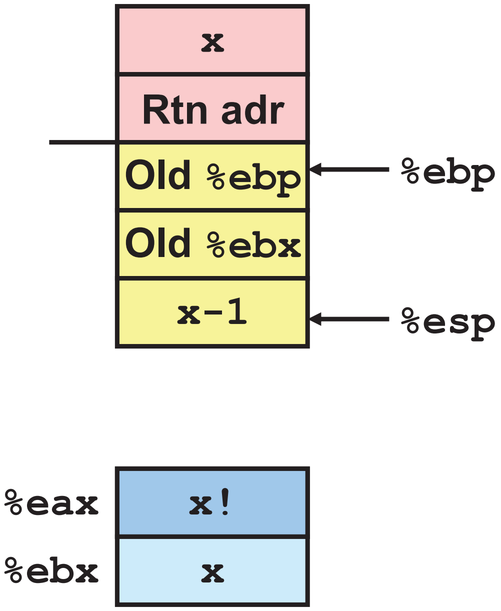


Return from Call

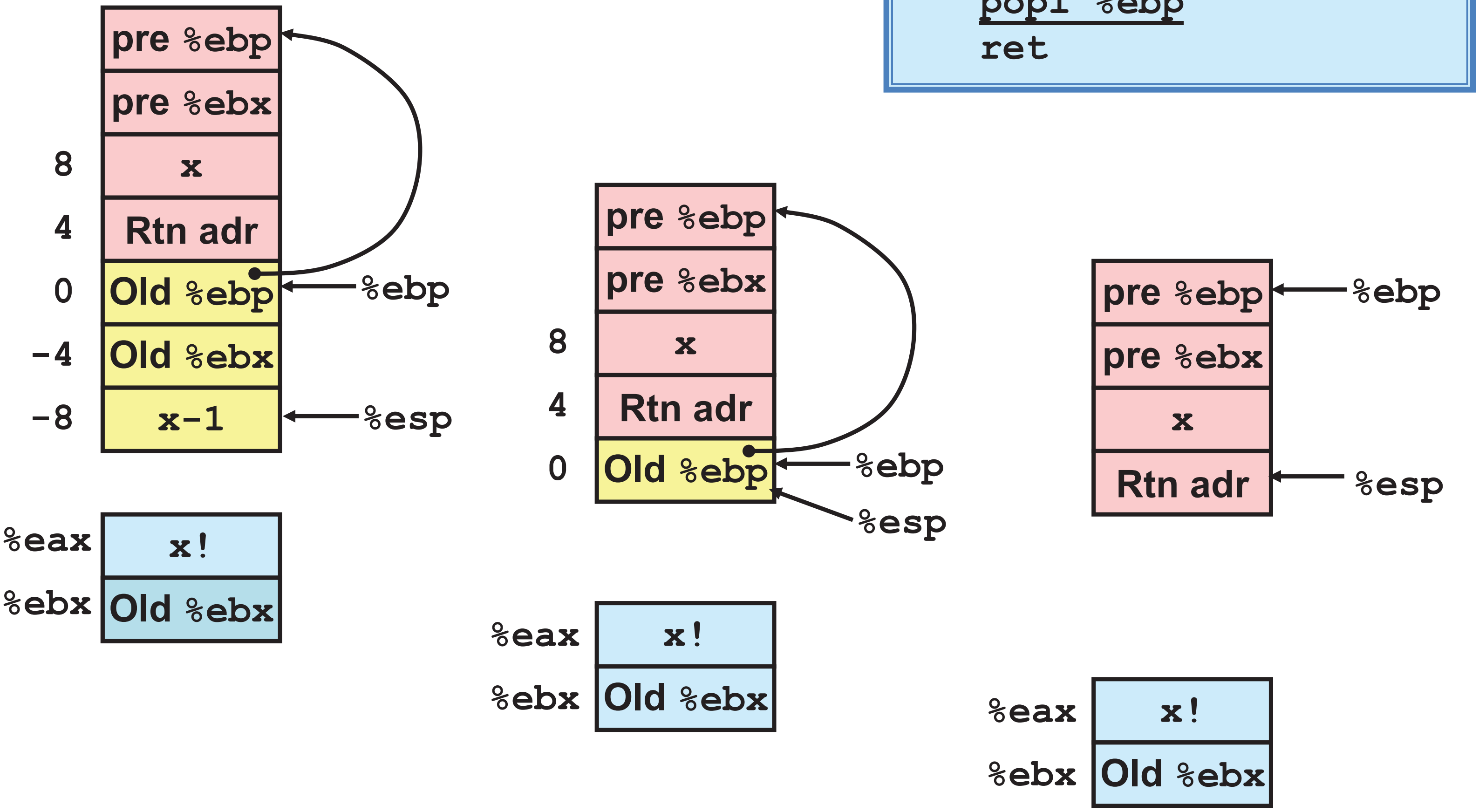


Assume that `rfact(x-1)` returns `(x-1)!` in register `%eax`

```
imull %ebx,%eax
```



```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```


pre %ebp
pre %ebx
8
x
4
Rtn adr
0
Old %ebp
%ebp
-4
Old %ebx
-8
x-1
%esp
%eax
x!
%ebx
Old %ebx
pre %ebp
pre %ebx
8
x
4
Rtn adr
0
Old %ebp
%ebp
%esp
%eax
x!
%ebx
Old %ebx
pre %ebp
%ebp
pre %ebx
x
Rtn adr
%esp
%eax
x!
%ebx
Old %ebx