Announcements

- Exam 2
  - Due Friday, October 24
  - Covers Lectures 9-18.
  - Format:
    - Multiple-Choice (30%)
    - Short-Answer (70%)
- Assignment 4
  - Due 5 p.m., Monday, October 27
- Graded lab next week
  - Exercises based on Assignments 2 & 3

Lecture 19

Recursive Procedures

CPSC 275
Introduction to Computer Systems

Recursive procedures

- Run-time stack allows procedures to call themselves recursively
- local variables of the multiple outstanding calls do not interfere with one another
- local storage allocated when the procedure is called and deallocated when it returns

The `factorial` function

```
void main() {
   printf("%d", rfact(4));
}
```

```
.LC0:
   .string  "%d\n"

main:
   pushl %ebp
   movl  %esp,%ebp

   pushl $4
   call  rfact
   pushl %eax
   pushl $.LC0
   call  printf

   leave
   ret
```

The `factorial` function

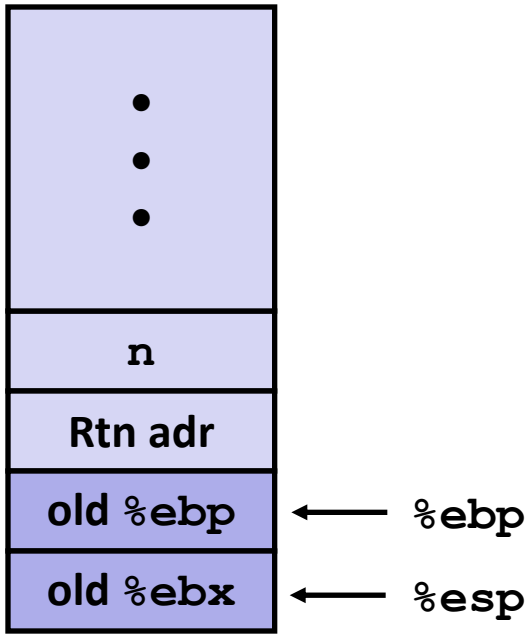
```
void main() {
   printf("%d", rfact(4));
}

int rfact(int n) {
   int result;
   if (n <= 1)
      result = 1;
   else
      result = n * rfact(n-1);
   return result;
}
```

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

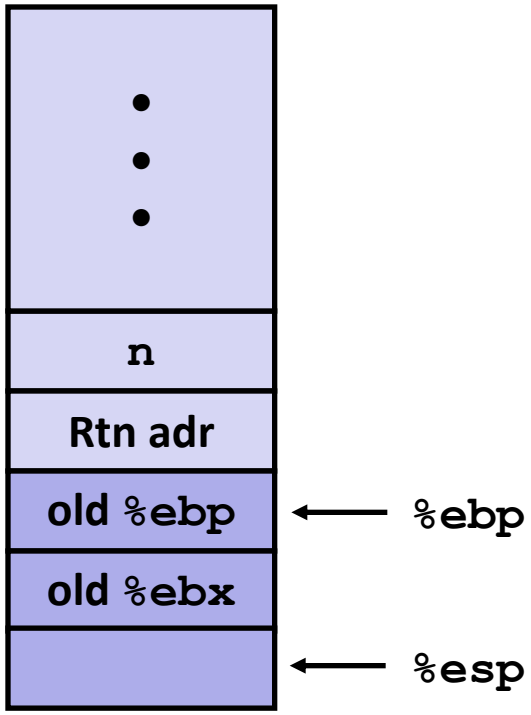
Stack frame just before the recursive call



The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

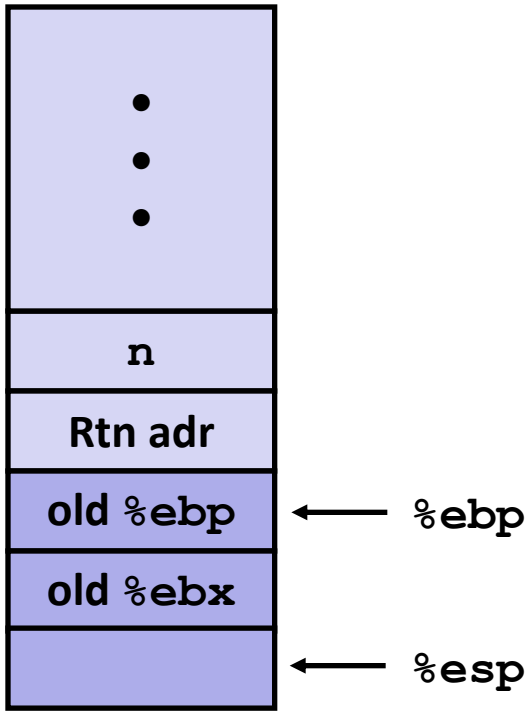
Stack frame just before the recursive call



The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

Stack frame just before the recursive call

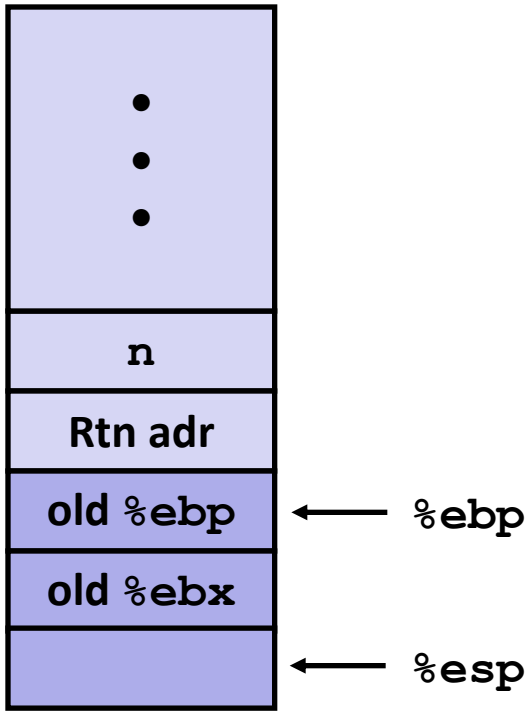


`%ebx = n`

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

Stack frame just before the recursive call



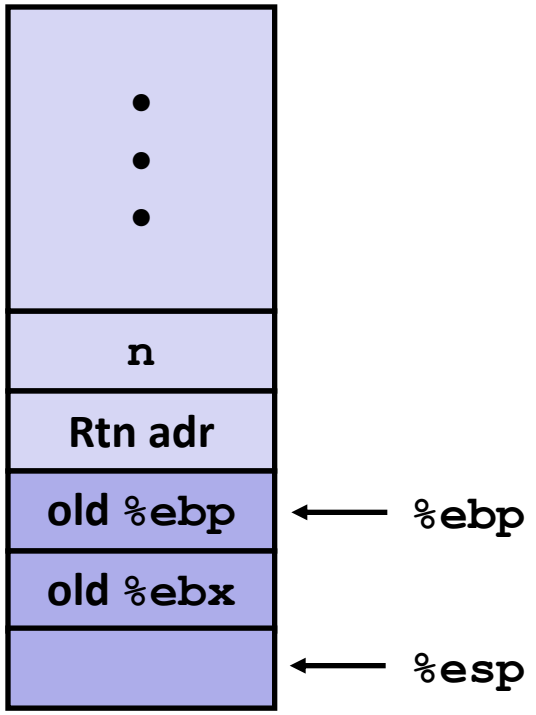
`%ebx = n`
`%eax = 1`

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

`%ebx == 1?`

Stack frame just before the recursive call



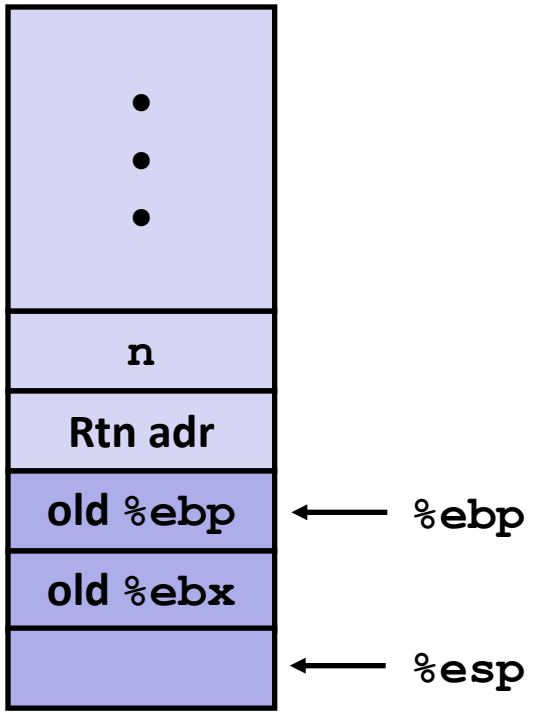
`%ebx = n`
`%eax = 1`

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

%ebx == 1? N

Stack frame just before the recursive call



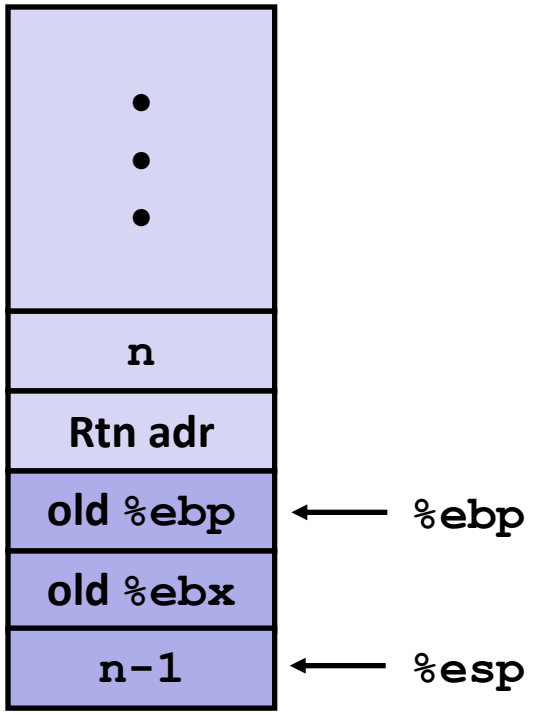
%ebx = n
%eax = n-1

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

%ebx == 1? N

Stack frame just before the recursive call



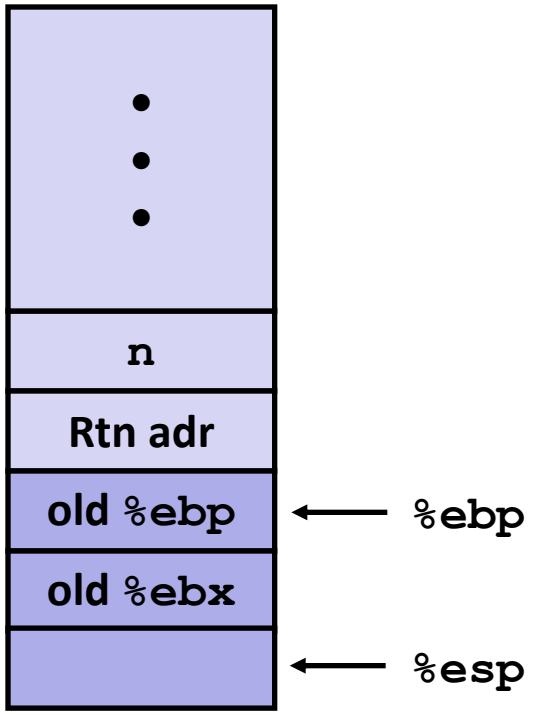
%ebx = n
%eax = n-1

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

%ebx == 1? Y

Stack frame just before the recursive call



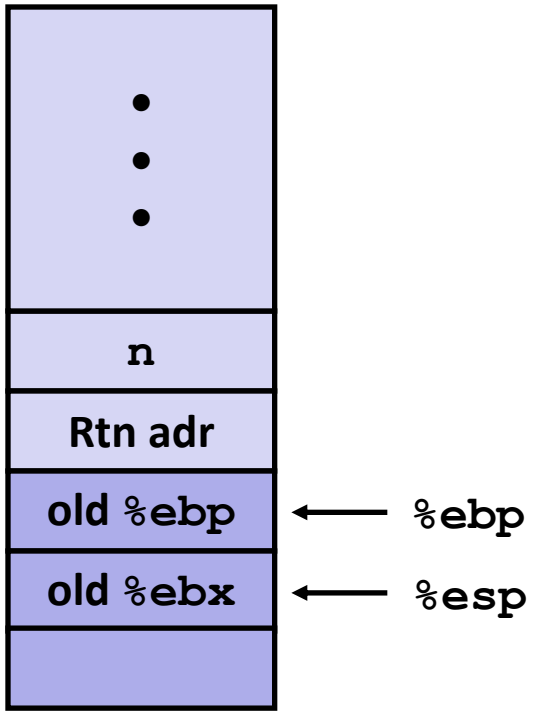
%ebx = n
%eax = 1

The `factorial` function

```
rfact:
    pushl   %ebp
    movl    %esp,%ebp
    pushl   %ebx
    subl    $4,%esp
    movl    8(%ebp),%ebx
    movl    $1,%eax
    cmpl    $1,%ebx
    jle     .done
    leal    -1(%ebx),%eax
    movl    %eax,(%esp)
    call    rfact
    imull   %ebx,%eax
.done:
    addl    $4,%esp
    popl    %ebx
    popl    %ebp
    ret
```

%ebx == 1? Y

Stack frame just before the recursive call



%ebx = n
%eax = 1