Machine-Level Programming III: Procedures

Appendix

Today

- Procedures
 - Stack Structure
 - Calling Conventions
 - Passing control
 - Passing data
 - Managing local data
 - Illustration of Recursion

Recursive Function

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
        .L6
 jе
 pushq %rbx
 movq %rdi, %rbx
 andl $1, %ebx
        %rdi
 shrq
 call
        pcount r
 addq %rbx, %rax
        %rbx
 popq
.L6:
 rep; ret
```

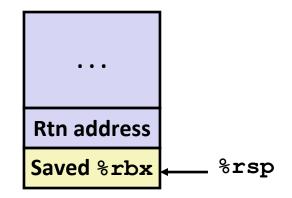
Recursive Function Terminal Case

pcount_r:	
movl	\$0, %eax
testq	%rdi, %rdi
je	.L6
pushq	%rbx
movq	%rdi, %rbx
andl	\$1, %ebx
shrq	%rdi
call	pcount_r
addq	%rbx, %rax
popq	%rbx
.L6:	
rep; re	t

```
RegisterUse(s)Type%rdixArgument%raxReturn valueReturn value
```

Recursive Function Register Save

```
pcount r:
 movl
         $0, %eax
        %rdi, %rdi
  testq
         .L6
  jе
 pushq
        %rbx
 movq %rdi, %rbx
 andl $1, %ebx
  shrq %rdi
 call
        pcount r
 addq %rbx, %rax
         %rbx
 popq
.L6:
 rep; ret
```



Register	Use(s)	Туре
%rdi	x	Argument

Recursive Function Call Setup

```
/* Recursive popcount */
long pcount_r(unsigned long x) {
 if (x == 0)
    return 0;
 else
    return (x & 1)
           + pcount_r(x >> 1);
```

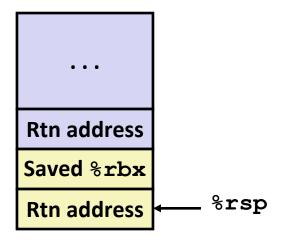
<pre>pcount_r:</pre>		• • •	
movl	\$0, %eax		
testq	%rdi, %rdi	Rtn address	
je	.L6	Carrad 0 1	0
pushq	%rbx	Saved %rbx	← %rsp
movq	%rdi, %rbx	,	,
andl	\$1, %ebx	%rbx = 2	s & 1
shrq	%rdi	L	
call	pcount_r		
addq	%rbx, %rax		
popq	%rbx		
.L6:			
rep; re	t		

```
Register
          Use(s)
                            Type
%rdi
          x >> 1
                            Rec. argument
                            Callee-saved
%rbx
         x & 1
```

Recursive Function Call

```
movl $0, %eax
 testq %rdi, %rdi
        .L6
 je
       %rbx
 pushq
 movq %rdi, %rbx
 andl
        $1, %ebx
       %rdi
 shrq
 call pcount r
 addq
       %rbx, %rax
        %rbx
 popq
.L6:
 rep; ret
```

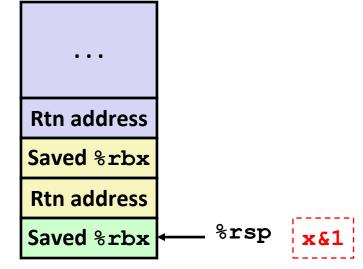
pcount r:



Register	Use(s)	Туре
%rbx	x & 1	Callee-saved
%rax	Recursive call return value	

Recursive Function Register Save #2

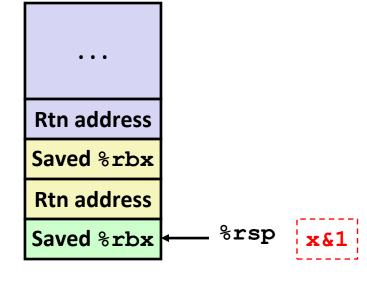
```
pcount r:
        $0, %eax
 movl
 testq
        %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
 movq %rdi, %rbx
 andl
         $1, %ebx
         %rdi
 shrq
 call pcount r
         %rbx, %rax
 addq
         %rbx
 popq
.L6:
 rep; ret
```



```
Register Use(s) Type
%rdi (x >> 1) Argument
```

Recursive Function Call Setup #2

```
pcount r:
         $0, %eax
 movl
 testq
         %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
        %rdi, %rbx
 movq
 andl
         $1, %ebx
         %rdi
 shrq
 call
         pcount r
 addq
         %rbx, %rax
         %rbx
 popq
.L6:
 rep; ret
```

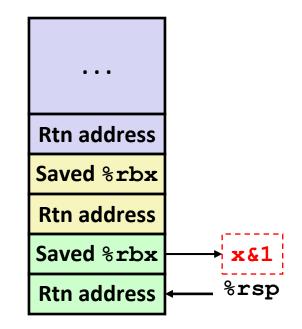


%rbx = (x >> 1) & 1

```
RegisterUse(s)Type%rdi(x >> 1) >> 1Rec. argument%rbx(x >> 1) & 1Callee-saved
```

Recursive Function Call #2

```
pcount r:
        $0, %eax
 movl
 testq
        %rdi, %rdi
        .L6
 jе
        %rbx
 pushq
        %rdi, %rbx
 movq
 andl
        $1, %ebx
        %rdi
 shrq
        pcount r
 call
 addq
        %rbx, %rax
        %rbx
 popq
.L6:
 rep; ret
```

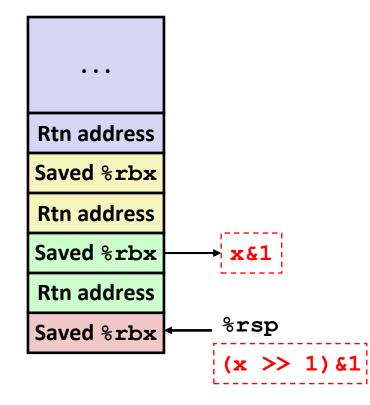


Register	Use(s)	Туре
%rbx	(x >> 1) & 1	Callee-saved
%rax	Recursive call return value	

Recursive Function Register Save #3

```
pcount r:
        $0, %eax
 movl
 testq
        %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
 movq %rdi, %rbx
 andl
        $1, %ebx
 shrq
        %rdi
 call pcount r
 addq
        %rbx, %rax
         %rbx
 popq
.L6:
```

rep; ret



```
Register Use(s) Type
%rdi (x >> 1) >> 1 Argument
```

Recursive Function Call Setup #3

```
/* Recursive popcount */
long pcount r(unsigned long x) {
  if (x == 0)
    return 0;
 else
    return (x & 1)
           + pcount r(x >> 1);
```

```
pcount r:
        $0, %eax
 movl
 testa
        %rdi, %rdi
         .L6
 jе
 pushq
        %rbx
 movq %rdi, %rbx
         $1, %ebx
 andl
         %rdi
 shrq
 call pcount r
 addq
         %rbx, %rax
         %rbx
 popq
.L6:
 rep; ret
```

```
Rtn address
Saved %rbx
Rtn address
Saved %rbx
Rtn address
                %rsp
Saved %rbx
               (x >> 1) &1
```

```
Register | Use(s)
                            Type
%rdi
         ((x>>1)>>1) >> 1
                             Rec. argument
         ((x>>1)>>1) & 1
%rbx
                            Callee-saved
```

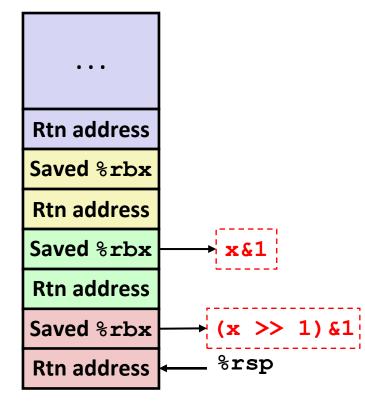
```
%rbx = ((x>>1)>>1) & 1
```

Recursive Function Call #3

```
pcount r:
         $0, %eax
 movl
 testq
        %rdi, %rdi
         .L6
 jе
         %rbx
 pushq
        %rdi, %rbx
 movq
 andl
         $1, %ebx
         %rdi
 shrq
         pcount r
 call
 addq
         %rbx, %rax
         %rbx
 popq
```

.L6:

rep; ret



```
Register Use(s)

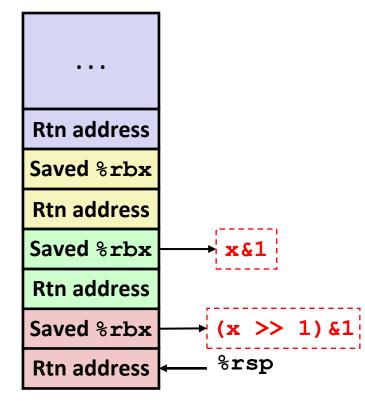
%rbx ((x>>1)>>1) & 1 Callee-saved

%rax Recursive call return value
```

Recursive Function #4 Terminal Case

```
pcount r:
         $0, %eax
 movl
 testq
        %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
        %rdi, %rbx
 movq
 andl
         $1, %ebx
         %rdi
 shrq
 call
         pcount r
 addq
         %rbx, %rax
         %rbx
 popq
.L6:
```

rep; ret



```
        Register
        Use(s)
        Type

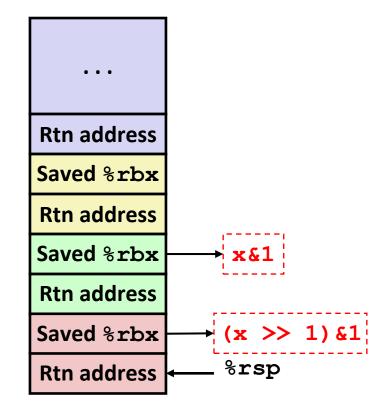
        %rdi
        ((x>>1)>>1) >> 1
        Argument

        %rbx
        ((x>>1)>>1) & 1
        Callee-saved
```

Recursive Function Return #4

```
pcount r:
         $0, %eax
 movl
 testq
        %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
        %rdi, %rbx
 movq
 andl
         $1, %ebx
         %rdi
 shrq
 call pcount r
 addq
         %rbx, %rax
         %rbx
 popq
.L6:
```

rep; ret



```
        Register
        Use(s)
        Type

        %rdi
        ((x>>1)>>1) >> 1
        Argument

        %rbx
        ((x>>1)>>1) & 1
        Callee-saved
```

Pop address from stack

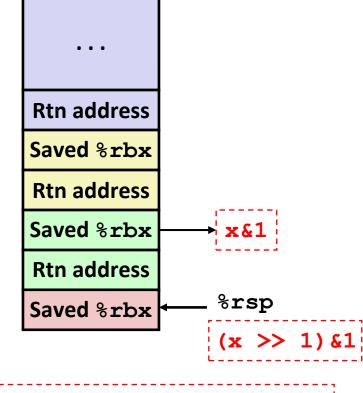
Jump to address

Recursive Function Add #3

```
pcount r:
        $0, %eax
 movl
 testa
        %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
        %rdi, %rbx
 movq
 andl
         $1, %ebx
        %rdi
 shrq
 call pcount r
        %rbx, %rax
 addq
         %rbx
 popq
```

.L6:

rep; ret



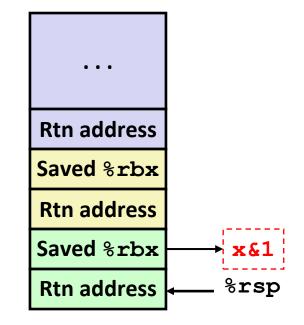
```
RegisterUse(s)Type%rbx((x>>1)>>1) & 1Callee-saved%rax0Recursive call return value
```

```
%rax = ((x>>1)>>1)& 1 + 0
```

Recursive Function Pre-return #3

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
        .L6
 je
 pushq %rbx
 movq %rdi, %rbx
 andl $1, %ebx
       %rdi
 shrq
 call pcount r
       %rbx, %rax
 addq
        %rbx
 popq
.L6:
```

rep; ret



Register	Use(s)	Туре
%rbx	(x >> 1) &1	Callee-saved
%rax	((x>>1)>>1)& 1	Recursive call return value

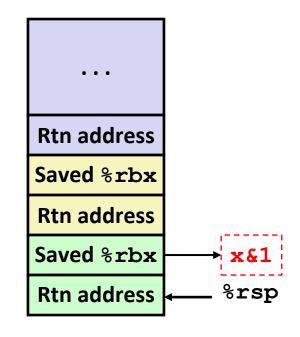
Restor %rbx

Recursive Function Return #3

pcount_r:	
movl	\$0, %eax
testq	%rdi, %rdi
je	.L6
pushq	%rbx
movq	%rdi, %rbx
andl	\$1, %ebx
shrq	%rdi
call	pcount_r
addq	%rbx, %rax
popq	%rbx

.L6:

rep; ret



```
RegisterUse(s)Type%rbx(x >> 1) &1Callee-saved%rax((x>>1)>>1) & 1Recursive call return value
```

Pop address from stack

Jump to address

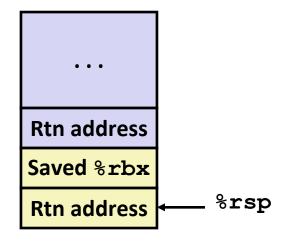
Recursive Function Add #2

Register	Use(s)	Туре
%rbx	(x >> 1) &1	Callee-saved
%rax	((x>>1)>>1)& 1	Recursive call return value

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
                        Rtn address
        .L6
 je
                        Saved %rbx
 pushq %rbx
 movq %rdi, %rbx
                        Rtn address
 andl $1, %ebx
                        Saved %rbx
                                   _%rsp
 shrq %rdi
 call pcount r
 addq %rbx, %rax
         %rbx
 popq
.L6:
                  % rax = (x >> 1) &1 +
 rep; ret
                        ((x>>1)>>1) & 1 + 0
```

Recursive Function Pre-return #2

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
        .L6
 je
 pushq %rbx
 movq %rdi, %rbx
 andl $1, %ebx
 shrq %rdi
 call pcount r
 addq %rbx, %rax
        %rbx
 popq
.L6:
 rep; ret
```

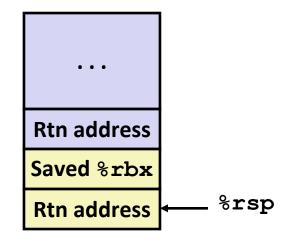


Register	Use(s)	Туре
%rbx	x & 1	Callee-saved
%rax	(x >> 1)&1 + ((x>>1)>>1)& 1 + 0	Recursive call return value

Restor %rbx

Recursive Function Return #2

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
        .L6
 jе
 pushq %rbx
 movq %rdi, %rbx
 andl $1, %ebx
        %rdi
 shrq
 call pcount r
 addq
       %rbx, %rax
        %rbx
 popq
.L6:
 rep; ret
```



Register	Use(s)	Туре
%rbx	x & 1	Callee-saved
%rax	(x >> 1) &1 + ((x>>1) >> 1) & 1 + 0	Recursive call return value

Pop address from stack

Jump to address

Recursive Function Add #1

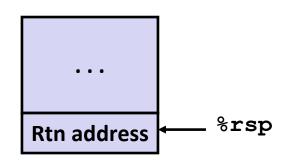
Register	Use(s)	Туре
%rbx	x & 1	Callee-saved
%rax	(x >> 1) &1 + ((x>>1) >> 1) & 1 + 0	Recursive call return value

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
                        Rtn address
         .L6
 je
                                    %rsp
                        Saved %rbx
 pushq %rbx
 movq %rdi, %rbx
 andl $1, %ebx
        %rdi
 shrq
 call pcount r
 addq %rbx, %rax
         %rbx
 popq
.L6:
                   %rax = (x & 1) +
 rep; ret
                         (x >> 1) &1 +
                         ((x>>1)>>1) & 1 + 0
```

Recursive Function Pre-return #1

```
pcount r:
 movl $0, %eax
 testq %rdi, %rdi
        .L6
 jе
 pushq %rbx
 movq %rdi, %rbx
 andl $1, %ebx
        %rdi
 shrq
 call pcount r
 addq
       %rbx, %rax
        %rbx
 popq
.L6:
```

rep; ret



```
Register Use(s)

%rbx

Callee-saved value

Callee-saved

Recursive call

(x >> 1) &1 +

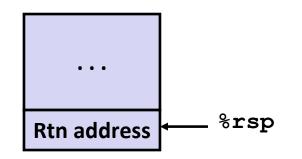
((x>>1) >> 1) & 1 +

0
```

Restor %rbx

Recursive Function Return #1 (Final)

```
pcount r:
        $0, %eax
 movl
 testq
        %rdi, %rdi
         .L6
 jе
        %rbx
 pushq
        %rdi, %rbx
 movq
 andl
         $1, %ebx
         %rdi
 shrq
 call pcount r
 addq
         %rbx, %rax
         %rbx
 popq
.L6:
 rep; ret
```



```
Register Use(s)

%rbx
Callee-saved value
Callee-saved

%rax
(x & 1) +
(x >> 1) & 1 +
((x>>1) >> 1) & 1 +
0

Type

Callee-saved
Recursive call return value
```

Pop address from stack

Jump to address

Recursive Function Call Stack

