Recursive Procedure Calls

Young W. Lim

2020-08-24 Tue

Outline

Based on

1. "Self-service Linux: Mastering the Art of Problem Determination", Mark Wilding

1. "Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Compling 32-bit program on 64-bit gcc

- gcc -v
- gcc -m32 t.c
- sudo apt-get install gcc-multilib
- sudo apt-get install g++-multilib
- gcc-multilib
- g++-multilib
- gcc -m32
- objdump -m i386

TOC: Recursive Procedure Example

- Recursive procedure
- Fibonacci sequence definition
- F_5 calling sequence example
- F_5 stack frame sequence
- Stack frame for $F_n = F_{n-2} + F_{n-1}$

Recurive procedure

- each call has its own private space on the stack
  - the local variables of the multiple outstanding calls do not interfere with one another
- the stack discipline naturally provides the proper policy
  - allocating local storage when the procedure is called
  - deallocating it when it returns

Fibonacci sequence definition (1)

- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$

n	0	1	2	3	4	5	6	7	8	9	10	11	12	...
F_n	0	1	1	2	3	5	8	13	21	34	55	89	144	...

- example)

$$\begin{aligned} F_5 &= F_4 + F_3 \\ &= (F_3 + F_2) + (F_2 + F_1) \\ &= ((F_2 + F_1) + (F_1 + F_0)) + ((F_1 + F_0) + F_1) \end{aligned} \tag{1}$$

Fibonacci sequence definition (2)

returning when ($n \leq 2$)

- $F_2 = 1$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$

n	1	2	3	4	5	6	7	8	9	10	11	12	...
F_n	1	1	2	3	5	8	13	21	34	55	89	144	...

pseudo code

- `fibo(n) = 1` $(n \leq 2)$
- `fibo(n) = fibo(n-2) + fibo(n-1)`

F_5 calling sequence example

1.	fibo(5)			
2.		fibo(3)		
3.			fibo(1)	
4.			fibo(2)	
5.		fibo(4)		
6.			fibo(2)	
7.			fibo(3)	
8.				fibo(1)
9.				fibo(2)

F_5 stack frame sequence (1)

push 5	push 3	push 1	pop 1	push 2	pop 2
fibo(5)	fibo(5)	fibo(5)	fibo(5)	fibo(5)	fibo(5)
	fibo(3)	fibo(3)	fibo(3)	fibo(3)	fibo(3)
		fibo(1)		fibo(2)	

- push n : calling fibo(n)
  allocate a stack frame for fibo(n)
- pop n : returning fibo(n)
  deallocate a stack frame for fibo(n)

F_5 stack frame sequence (2)

pop 3	push 4	push 2	pop 2	push 3	push 1
fibo(5)	fibo(5)	fibo(5)	fibo(5)	fibo(5)	fibo(5)
	fibo(4)	fibo(4)	fibo(4)	fibo(4)	fibo(4)
		fibo(2)		fibo(3)	fibo(3)
					fibo(1)

- push n : calling fibo(n)
  allocate a stack frame for fibo(n)
- pop n : returning fibo(n)
  deallocate a stack frame for fibo(n)

F_5 stack frame sequence (3)

pop 1	push 2	pop 2	pop 3	pop 4	pop 5
fibo(5)	fibo(5)	fibo(5)	fibo(5)	fibo(5)	
fibo(4)	fibo(4)	fibo(4)	fibo(4)		
fibo(3)	fibo(3)	fibo(3)			
	fibo(2)				

- push n : calling fibo(n)
  allocate a stack frame for fibo(n)
- pop n : returning fibo(n)
  deallocate a stack frame for fibo(n)

Stack frames for $F_n = F_{n-2} + F_{n-1}$

push n	push n-2	::	pop n-2	push n-1	::	pop n-1	pop n
fibo(n)	fibo(n)	::	fibo(n)	fibo(n)	::	fibo(n)	
	fibo(n-2)	::		fibo(n-1)	::		
		::			::		

- push n : calling fibo(n)
  allocate a stack frame for fibo(n)
- pop n : returning fibo(n)
  deallocate a stack frame for fibo(n)

TOC: fibo codes

- Fibonacci source code
- Fibonacci assembly code (1) body
- Fibonacci assembly code (2) setup
- Fibonacci assembly code (3) finish

Fibonacci source code

Fibonacci source code

```
int fibo(int n) {
  int prev, val;

  if (n <= 2) return 1;
  //------------------------
  prev = fibo(n-2);
  //------------------------
  val  = fibo(n-1);
  //------------------------
  return prev + val;
}
```

- for the input argument `n`, `%ebx` will be used
- to compute the result `prev + res` , `%esi` will be used
- old values of `%ebx` and `%esi` will be saved on the stack

Fibonacci assembly code (1) body

Fibonacci Body code

```
;-------------------------
movl   8(%ebp), %ebx
cmpl   $2,      %ebx
jle    .L24
;-------------------------
addl   $-12,    %esp
leal   -2(%ebx), %eax
pushl  %eax
call   fibo
movl   %eax,    %esi
;-------------------------
addl   $-12,    %esp
leal   -1(%ebx), %eax
pushl  %eax
call   fibo
addl   %esi,    %eax
;-------------------------
jmp    .L25
;-------------------------
```

- if the parameter $n \leq 2$ then termination condition
- the 1st recursive call
  - the value (n-2) is pushed
  - call fibo(n-2)
  - save the result in %esi
  - 4 words (16 bytes) stack frame allocated
- the 2nd recursive call
  - the value (n-1) is pushed
  - call fibo(n-1)
  - add the result into %esi
  - 4 words (16 bytes) stack frame allocated

Fibonacci assembly code (2) setup

Fibonacci Setup code

```
; setup code
fibo:
  pushl %ebp
  movl  %esp, %ebp
  subl  $16, %esp
  pushl %esi
  pushl %ebx
```

- %ebp, %esi, %ebx are pushed onto the stack
- 7 words (28 bytes) stack frame allocated

Fibonacci assembly code (3) finish

Fibonacci Finish code

```
; terminal condition
.L24:
movl $1 %eax

; finishing code
.L25:
leal -24(%ebp),%esp
popl %ebx
popl %esi
movl %ebp, %esp
popl %ebp
ret
```

- if terminal condition,
  set the return value to 1
- deallocates two
  4 words (16 bytes) that was
  allocated in the body code
- deallocates
  7 words (28 bytes) that was
  allocated in the setup code
  holding the parameter "n"
- %ebx, %esi, %ebp are
  popped from the stack

TOC: 1. Setup code for `fibo`

- Setup code summary
- Setup code stack frame
- Setup code and the input parameter `n`

Setup code summary

setup code

```
fibo:                   ; Set up code
  pushl %ebp            ; push frame pointer %ebp   ; %ebp -> M[%ebp- 0]
  movl  %esp, %ebp      ; new frame pointer %ebp    ; %esp -> %ebp
  subl  $16, %esp       ; alloc 16 bytes on stack   ; %esp - 16 -> %esp
  pushl %esi            ; push %esi                 ; %esi -> M[%ebp-20]
  pushl %ebx            ; push %ebx                 ; %ebx -> M[%ebp-24]
```

- creates a 7 words (28 bytes) stack frame
  - the old %ebp at %ebp+0
  - 16 unused bytes from %ebp-4, -8, -12, -16
  - callee save register %esi at %ebp-20
  - callee save register %ebx at %ebp-24
- %esi is used for computing the return value of fibo
- %ebx holds the procedure parameter n

Setup code stack frame (1)

a) pushl %ebp

```
        |                          |
        |---------+-------------|
%esp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 |             |6
        | %ebp-24 |             |7
        |---------+-------------|

        * pushl %ebp
          movl  %esp, %ebp
          subl  $16, %esp
          pushl %esi
          pushl %ebx
```

b) movl %esp %ebp

```
        |                          |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 |             |6
        | %ebp-24 |             |7
        |---------+-------------|

        * pushl %ebp
        * movl  %esp, %ebp
          subl  $16, %esp
          pushl %esi
          pushl %ebx
```

Setup code stack frame (2)

c) subl $16, %esp

```
        |                          |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
%esp -> | %ebp-16 |             |5
        | %ebp-20 |             |6
        | %ebp-24 |             |7
        |---------+-------------|
```

* pushl %ebp
* movl %esp, %ebp
* subl $16, %esp

  pushl %esi

  pushl %ebx

d) pushl %esi, pushl %ebx

```
        |                          |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
%esp -> | %ebp-24 | saved %ebx  |7
        |---------+-------------|
```

* pushl %ebp
* movl %esp, %ebp
* subl $16, %esp
* pushl %esi
* pushl %ebx

Setup code and the input parameter n

setup code

```
fibo:
  pushl %ebp          ; 1st word
  movl  %esp, %ebp    ; update %ebp
  subl  $16, %esp     ; 2nd - 5th word
  pushl %esi          ; 6th word
  pushl %ebx          ; 7th word

  ; %esi : used for computing the
  ;        return value
  ; %ebx : the procedure parameter n
  ;        on the stack (M[%ebp+8])
  ;        that the caller pushed
  ;        before calling fibo
```

input parameter n

```
          |stack frame for caller |
          |vvvvvvvvv+vvvvvvvvvvvvv|
        * | %ebp+8   | n           |
          | %ebp+4   | return addr |
          |----------+-------------|
%ebp ->   | %ebp+0   | saved %ebp  |1
          | %ebp-4   |             |2
          | %ebp-8   |             |3
          | %ebp-12  |             |4
          | %ebp-16  |             |5
          | %ebp-20  | saved %esi  |6
%esp ->   | %ebp-24  | saved %ebx  |7
          |----------+-------------|
          |stack frame for fibo    |
```

TOC: 2. Body code for `fibo`

- Body code summary
- Body code stack frame
- Passing the parameter $n-2$ and $n-1$
- Computing the return value
- Each call recursively triggers
- Frist recursive call to `fibo`
- Second recursive call to `fibo`
- Stack frame just before the 1st `fibo` call
- Stack frame just before the 2nd `fibo` call

Body code summary

body code

```
;-------------------------------------------------------------------
movl   8(%ebp), %ebx  ; get n                          ; M[%ebp+8] -> %ebx
cmpl   $2,      %ebx  ; compare n == 2                 ; %ebx -2
jle    .L24           ; if <=, goto terminate          ; goto .L24
;-------------------------------------------------------------------
addl   $-12,    %esp  ; allocate 12 bytes on stack     ; %esp -12 -> %esp
leal  -2(%ebx), %eax  ; compute (n-2)                  ; %ebx -2 -> %eax
pushl  %eax           ; push the argument (n-2)        ; %eax -> M[%ebp-40]
call   fibo           ; call fibo(n-2)                 ; setup-body-finish
movl   %eax,    %esi  ; store result in %esi           ; %eax -> %esi
;-------------------------------------------------------------------
addl   $-12,    %esp  ; allocate 12 tyes on stack      ; %esp -12 -> %esp
leal  -1(%ebx), %eax  ; compute (n-1)                  ; %ebx -1 -> %eax
pushl  %eax           ; push the argument (n-1)        ; %eax -> M[%ebp-56]
call   fibo           ; call fibo(n-1)                 ; setup-body-finish
addl   %esi,    %eax  ; compute val+nval               ; %eax +%esi -> %eax
;-------------------------------------------------------------------
jmp    .L25           ; goto done
```

Body code stack frame (1) body1

a) addl $-12. %esp

```
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        |.....................|
        | %ebp-28 |             |8
        | %ebp-32 |             |9
%esp -> | %ebp-36 |             |10
        | %ebp-40 |             |11
        |---------+-------------|

        * addl   $-12,    %esp
          pushl  %eax
```

b) pushl %eax

```
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        |.....................|
        | %ebp-28 |             |8
        | %ebp-32 |             |9
        | %ebp-36 |             |10
%esp -> | %ebp-40 | n-2 **      |11
        |---------+-------------|

        * addl   $-12,    %esp
        * pushl  %eax
```

Body code stack frame (2) body2

c) addl $-12. %esp

```
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        |.....................|
        | %ebp-28 |             |8
        | %ebp-32 |             |9
        | %ebp-36 |             |10
        | %ebp-40 | n-2         |11
        |.....................|
        | %ebp-44 |             |12
        | %ebp-48 |             |13
%esp -> | %ebp-52 |             |14
        | %ebp-56 |             |15
        |---------+-------------|
       * addl   $-12,    %esp
         pushl  %eax
```

d) pushl %eax

```
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        |.....................|
        | %ebp-28 |             |8
        | %ebp-32 |             |9
        | %ebp-36 |             |10
        | %ebp-40 | n-2         |11
        |.....................|
        | %ebp-44 |             |12
        | %ebp-48 |             |13
        | %ebp-52 |             |14
%esp -> | %ebp-56 | n-1 **      |15
        |---------+-------------|
       * addl   $-12,    %esp
       * pushl  %eax
```

Passing the parameter $n-2$ and $n-1$

a parameter passing

```
movl   8(%ebp), %ebx   ; get n                          ; M[%ebp+8] -> %ebx
;.....................................................................
leal  -2(%ebx), %eax   ; compute n-2                    ; %ebx -2 -> %eax
pushl  %eax            ; push the argument (n-2)
;.....................................................................
leal  -1(%ebx), %eax   ; compute n-1                    ; %ebx -1 -> %eax
pushl  %eax            ; push the argument (n-1)
```

- register %ebx contains the value of function argument n
- for the 1st recursive call fibo (n-2)
  compute the only arugment (n-2)
  push the value unto the stack
- for the 2nd recursive call fibo (n-1)
  compute the only arugment (n-1)
  push the value unto the stack

Computing the return value

return value

```
;.....................................................................
call    fibo
movl    %eax,     %esi     ; %eax -> %esi
;.....................................................................
call    fibo
addl    %esi,     %eax     ; %eax + %esi -> %eax
```

- register %eax contains the value returned by the recursive call
- save the return value of the 1st recursive call into %esi
- add %esi with the return value %eax of the 2nd recursive call and store the result into %eax
- in the event of a terminal condition, the code jumps to line 22, where the return value is set to 1

Each call recursively triggers

- the recursive call will trigger a number of calls
  - `fibo(4)` calls
    - `fibo(2)` and
    - `fibo(3)`
  - `fibo(3)` calls
    - `fibo(1)` and
    - `fibo(2)`
- each call allocates its own stack frame
  performs operations on its own local storage
- as each call returns, it deallocates any stack space and
  restores any modified callee save registers

First recursive call to fibo

1st recursive call

```
addl   $-12,    %esp      ; allocate 12 bytes on stack  ; %esp -12 -> %esp
leal   -2(%ebx), %eax     ; compute n-2                 ; %ebx  -2 -> %eax
pushl  %eax               ; push an argument
call   fibo               ; call fibo(n-2)
```

- the stack grows by 4 words
  - allocating 12 bytes (3 words) that are never used
  - pushing the value "(n-2)"
- the return value in %eax

Second recursively call to fibo

2nd recursive call

```
addl   $-12,    %esp      ; allocate 12 tyes on stack  ; %esp -12 -> %esp
leal   -1(%ebx), %eax     ; compute n-1                ; %ebx  -2 -> %eax
pushl  %eax               ; push an argument
call   fibo               ; call fibo(n-1)
```

- the stack grows by 4 words
  - allocating 12 bytes (3 words) that are never used
  - pushing the value "(n-1)"
- the return value in %eax

Stack frame just before the 1st fibo call

1st fibo call

```
;-----------------------------------
addl   $-12,    %esp   ; 8th - 10th
leal   -2(%ebx), %eax  ; get n-2
pushl  %eax            ; push n-2
call   fibo
movl   %eax,    %esi   ; saved result
;-----------------------------------
```

just before the 1st fibo call

```
          |---------+-------------|
%ebp ->   | %ebp+0  | saved %ebp  |1
          | %ebp-4  |             |2
          | %ebp-8  |             |3
          | %ebp-12 |             |4
          | %ebp-16 |             |5
          | %ebp-20 | saved %esi  |6
          | %ebp-24 | saved %ebx  |7
          |.....................|
          | %ebp-28 |             |8
          | %ebp-32 |             |9
          | %ebp-36 |             |10
%esp ->   | %ebp-40 | n-2 **      |11
          |---------+-------------|
```

Stack frame just before the 2nd fibo call

2nd fibo call

```
;-----------------------------------
addl   $-12,    %esp   ; 8th - 10th
leal   -2(%ebx), %eax  ; get n-2
pushl  %eax            ; push n-2
call   fibo
movl   %eax,    %esi   ; saved result
;-----------------------------------
addl   $-12,    %esp   ; 12th - 14th
leal   -2(%ebx), %eax  ; get n-1
pushl  %eax            ; push n-1
call   fibo
addl   %eax,    %esi   ; added result
;-----------------------------------
```

just before the 2nd fibo call

```
          |---------+-------------|
%ebp ->   | %ebp+0  | saved %ebp  |1
          | %ebp-4  |             |2
          | %ebp-8  |             |3
          | %ebp-12 |             |4
          | %ebp-16 |             |5
          | %ebp-20 | saved %esi  |6
          | %ebp-24 | saved %ebx  |7
          |.....................|
          | %ebp-28 |             |8
          | %ebp-32 |             |9
          | %ebp-36 |             |10
          | %ebp-40 | n-2         |11
          |.....................|
          | %ebp-44 |             |12
          | %ebp-48 |             |13
          | %ebp-52 |             |14
%esp ->   | %ebp-56 | n-1 **      |15
          |---------+-------------|
```

TOC: Finish code

- Finish code summary
- Finish code stack frame
- Jump to `.L24`
- Jump to `.L25`
- Deallocating body code stack words
- Deallocating setup code stack words
- Setup code stack words
- Entering the 1st `fibo`
- Returning from the 1st `fibo`

Finish code summary

finish code

```
; terminal condition
.L24:                   ; terminate ; when n <= 2
  movl $1 %eax          ; return value 1

; finishing code
.L25:                   ; done  ; after returning from each recursive call
  leal -24(%ebp),%exp   ; set stack to offset -24
  popl %ebx             ; restore %ebx
  popl %esi             ; restore %esi
  movl %ebp, %esp       ; restore %esp
  popl %ebp             ; restore %ebp
  ret                   ; return to the caller
```

Finish code stack frame (1)

after the 2nd fibo call

```
         | %ebp+8   | n           |
         | %ebp+4   | return addr |
         |----------+-------------|
%ebp ->  | %ebp+0   | saved %ebp  |1
         | %ebp-4   |             |2
         | %ebp-8   |             |3
         | %ebp-12  |             |4
         | %ebp-16  |             |5
         | %ebp-20  | saved %esi  |6
         | %ebp-24  | saved %ebx  |7
         |.........................|
         | %ebp-28  |             |8
         | %ebp-32  |             |9
         | %ebp-36  |             |10
         | %ebp-40  | n-2         |11
         |.........................|
         | %ebp-44  |             |12
         | %ebp-48  |             |13
         | %ebp-52  |             |14
%esp ->  | %ebp-56  | n-1         |15
         |.........................|
```

after finish code

```
         | %ebp+8   | n           |
%esp ->  | %ebp+4   | return addr |
         |----------+-------------|
         : %ebp+0   : saved %ebp  :1
         : %ebp-4   :             :2
         : %ebp-8   :             :3
         : %ebp-12  :             :4
         : %ebp-16  :             :5
         : %ebp-20  : saved %esi  :6
         : %ebp-24  : saved %ebx  :7
         :.........................:
         : %ebp-28  :             :8
         : %ebp-32  :             :9
         : %ebp-36  :             :10
         : %ebp-40  : n-2         :11
         :.........................:
         : %ebp-44  :             :12
         : %ebp-48  :             :13
         : %ebp-52  :             :14
         : %ebp-56  : n-1 **      :15
         :.........................:
```

Finish code stack frame (2)

a) leal -24(%ebp), %esp

```
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |
        | %ebp-4  |             |
        | %ebp-8  |             |
        | %ebp-12 |             |
        | %ebp-16 |             |
        |.....................|
        | %ebp-20 | saved %esi  |
%esp -> | %ebp-24 | saved %ebx  |
        |---------+-------------|


        * leal -24(%ebp),%esp
          popl %ebx
          popl %esi
```

b) popl %ebx, popl %esi

```
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |
        | %ebp-4  |             |
        | %ebp-8  |             |
        | %ebp-12 |             |
%esp -> | %ebp-16 |             |
        |---------+-------------|
        : %ebp-20 : saved %esi  :
        : %ebp-24 : saved %ebx  :
        :                       :


        * popl %ebx
        * popl %esi
          movl %ebp, %esp
          popl %ebp
```

Finish code stack frame (3)

c) movl %ebp, %esp

```
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%esp -> | %ebp+0  | saved %ebp  |
        |---------+-------------|
        : %ebp-4  :             :
        : %ebp-8  :             :
        : %ebp-12 :             :
        : %ebp-16 :             :
        : %ebp-20 : saved %esi  :
        : %ebp-24 : saved %ebx  :

        * popl %ebx
        * popl %esi
        * movl %ebp, %esp
          popl %ebp
```

d) popl %ebp

```
        | %ebp+8  | n           |
%esp -> | %ebp+4  | return addr |
        |---------+-------------|
        : %ebp+0  : saved %ebp  :
        :.....................:
        : %ebp-4  :             :
        : %ebp-8  :             :
        : %ebp-12 :             :
        : %ebp-16 :             :
        : %ebp-20 : saved %esi  :
        : %ebp-24 : saved %ebx  :

        * popl %ebx
        * popl %esi
        * movl %ebp, %esp
        * popl %ebp
```

Jump to .L24

when to jump to .L24

```
;.......................
; get n
; M[%ebp+8] -> %ebx
;.......................
movl   8(%ebp), %ebx

;.......................
; compare n == 2
; %ebx -2
;.......................
cmpl   $2,      %ebx

;.......................
; if <=, goto terminate
;.......................
jle    .L24
```

terminating a recursive call

```
;.......................
; terminal condition
; terminate
;.......................
.L24:

;.......................
; return value 1
;.......................
movl $1 %eax

;.......................
; then do the finishing code
; to restore the stack
;.......................
.L25:
```

Jump to .L25

when to jump to .L25

```
;.......................
; 1st recursive call (n-2)
;.......................
call   fibo
movl   %eax,    %esi

;.......................
; 2nd recursive call (n-1)
;.......................
call   fibo
addl   %esi,    %eax

;.......................
; jump to finishing code
;.......................
jmp    .L25
```

restoring the stack

```
.L25:
;.......................
; shrink the stack by 6 words
;.......................
  leal -24(%ebp),%esp
;.......................
; restore %ebx, %esi
;.......................
  popl %ebx
  popl %esi
;.......................
; restore %esp, %ebp
;.......................
  movl %ebp, %esp
  popl %ebp
  ret
```

Deallocating body code stack words

shrink %esp

```
leal -24(%ebp),%esp ; set stack to offset -24
```

- deallocates 4 words which were allocated
  just before the recursive call in the body code
- starts by setting the stack frame
  to the location of the saved value of %ebx
  M[%ebp -24] = %ebx
- by computing this stack position
  relative to the value of %ebp,
  the computation will be correct
  regardless of whether or not the terminal condition
  ($n <= 2$) was reached

Deallocating setup code stack words

restore %ebx and %esi

```
popl %ebx              ; restore %ebx
popl %esi              ; restore %esi
```

- restore the saved values of %esi and %ebp
  %esi was used to store the result of the 1stecursive call
  %ebx was used as the procedure parameter n

restore %esp and %ebp

```
movl %ebp, %esp        ; restore %esp
popl %ebp              ; restore %ebp
```

- set %esp to where the old %ebp is stored
  then pop this value into the %ebp

Setup code stack words

stack setup and restore

```
fibo:
  pushl %ebp
  movl  %esp, %ebp
  subl  $16, %esp
  pushl %esi
  pushl %ebx


.L25:
  leal -24(%ebp),%esp
  popl %ebx
  popl %esi
  movl %ebp, %esp
  popl %ebp
  ret
```

setup code stack words

```
          |---------+-------------|
%ebp ->   | %ebp+0  | saved %ebp  |
          | %ebp-4  |             |
          | %ebp-8  |             |
          | %ebp-12 |             |
          | %ebp-16 |             |
          | %ebp-20 | saved %esi  |
          | %ebp-24 | saved %ebx  |
          |---------+-------------|
```

Entering the 1st fibo

before the 1st fibo

```
        |stack frame for caller |
        |---------+-------------|
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |
        | %ebp-4  |             |
        | %ebp-8  |             |
        | %ebp-12 |             |
        | %ebp-16 |             |
        | %ebp-20 | saved %esi  |
        | %ebp-24 | saved %ebx  |
        | %ebp-28 |             |
        | %ebp-32 |             |
        | %ebp-36 |             |
%esp -> | %ebp-40 | n-2         |
        |---------+-------------|
        |                       |
        |stack frame for fibo   |
```

entering the 1st fibo

```
        |stack frame for caller |
        |---------+-------------|
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        | %ebp-28 |             |8
        | %ebp-32 |             |9
        | %ebp-36 |             |10
        | %ebp-40 | n-2         |11
%esp -> | %ebp-40 | return addr |12
        |---------+-------------|
        |stack frame for fibo   |
```

Returning from the 1st fibo

entering the 1st fibo

```
        |stack frame for caller |
        |---------+-------------|
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        | %ebp-28 |             |8
        | %ebp-32 |             |9
        | %ebp-36 |             |10
        | %ebp-40 | n-2         |11
%esp -> | %ebp-40 | return addr |12
        |---------+-------------|
        |stack frame for fibo   |
```

returning from the 1st fibo

```
        |stack frame for caller |
        |---------+-------------|
        | %ebp+8  | n           |
        | %ebp+4  | return addr |
        |---------+-------------|
%ebp -> | %ebp+0  | saved %ebp  |1
        | %ebp-4  |             |2
        | %ebp-8  |             |3
        | %ebp-12 |             |4
        | %ebp-16 |             |5
        | %ebp-20 | saved %esi  |6
        | %ebp-24 | saved %ebx  |7
        | %ebp-28 |             |8
        | %ebp-32 |             |9
        | %ebp-36 |             |10
%esp -> | %ebp-40 | n-2         |11
        |---------+-------------|
        |                       |
        |stack frame for fibo   |
```

TOC: Stack frame tracing examples

- Code parts and affected stack frame parts
- 1. Setup code and `fibo(n)_7w`
- 2. Body code and `fibo(n)_4w`
- 3. Finish code and `fibo(n)_7w, 4w1, 4w2`
- Notation `fibo(n)_7w` and `fibo(n)_4w`
- F_n calling sequence in detail
- F_3 calling sequence
- F_3 calling sequence in detail
- Stack frames for `fibo(3)` computation
- F_4 calling sequence
- F_4 calling sequence in detail
- Stack frames for `fibo(4)` computation

Code parts and affected stack frame parts

code parts	affected stack frame parts	
setup code	create a stack frame for `fibo(n)`	`fibo(n)_7w`
body1 code	increase the stack frame for a recurisve call `fibo(n-2)`	`fibo(n)_4w1`
body2 code	increase the stack frame for another recursive call `fibo(n-1)`	`fibo(n)_4w2`
finish code	remove the stack frame for `fibo(n)`	

1. Setup code and fibo(n)_7w

fibo(n) setup code

```
; setup code
fibo:
  pushl %ebp            ; 1w
  movl  %esp, %ebp
  subl  $16, %esp       ; 4w
  pushl %esi            ; 1w
  pushl %ebx            ; 1w
```

fibo(n)_7w

```
          |---------+-------------|
%ebp ->   | %ebp+0  | saved %ebp  |1
          | %ebp-4  |             |2
          | %ebp-8  |             |3
          | %ebp-12 |             |4
          | %ebp-16 |             |5
          | %ebp-20 | saved %esi  |6
          | %ebp-24 | saved %ebx  |7
          |---------+-------------|
```

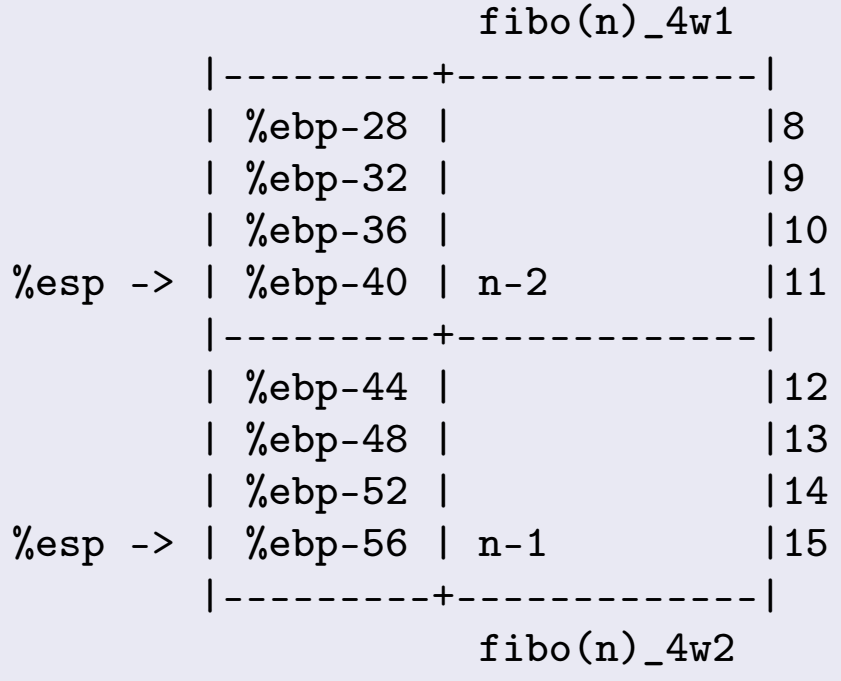
2. Body code and fibo(n)_4w

fibo(n) body code

```
;-------------------------
; body1 code
; expand 4w1
;-------------------------
addl   $-12,    %esp   ; 3w
leal   -2(%ebx), %eax
pushl  %eax            ; 1w

;-------------------------
; body2 code
; expand 4w2
;-------------------------
addl   $-12,    %esp   ; 3w
leal   -2(%ebx), %eax
pushl  %eax            ; 1w
```

fibo(n)_4w1, _4w2



3. Finish code and fibo(n)_7w, 4w1, 4w2

fibo(n) finish code

```
;---------------------
; finish1 code
; remove 4w1/4w2
;---------------------
leal -24(%ebp),%exp

;---------------------
; finish2 code
; remove 7w
;---------------------
popl %ebx
popl %esi
movl %ebp, %esp
popl %ebp
```

fibo(n)_7w, _4w1, _4w2

```
%esp -> |                          |
        |---------+--------------|
%ebp -> | %ebp+0  | saved %ebp   |1
        | %ebp-4  |              |2
        | %ebp-8  |              |3
        | %ebp-12 |              |4
        | %ebp-16 |              |5
        | %ebp-20 | saved %esi   |6
%esp -> | %ebp-24 | saved %ebx   |7
        |---------+--------------|
        | %ebp-28 |              |8
        | %ebp-32 |              |9
        | %ebp-36 |              |10
%esp -> | %ebp-40 | n-2          |11
        |---------+--------------|
        | %ebp-28 |              |12
        | %ebp-32 |              |13
        | %ebp-36 |              |14
%esp -> | %ebp-40 | n-1          |15
        |---------+--------------|
```

Notation fibo(n)_7w and fibo(n)_4w (1)

fibo(n)_7w	fibo(n)_4w1	fibo(n)_4w2
created by setup	created by body1	created by body2
7 word size	4 word size	4 word size
saved %ebp saved %esi saved %ebx	the argument (n-2)	the argument (n-1)
removed by finish2	removed by finish1	removed by finish1

Notation fibo(n)_7w and fibo(n)_4w (2)

fibo(n)_7w and fibo(n)_4w

```
         |stack frame for caller |
         |---------+-------------|
         | %ebp+8  | n           |
         | %ebp+4  | return addr |
         |---------+-------------|.............................
%ebp ->  | %ebp+0  | saved %ebp  |1   expanded by setup
         | %ebp-4  |             |2   removed by finish2
         | %ebp-8  |             |3
         | %ebp-12 |             |4
         | %ebp-16 |             |5   "fibo(n)_7w"
         | %ebp-20 | saved %esi  |6
         | %ebp-24 | saved %ebx  |7
         |.........................|.............................
         | %ebp-28 |             |8   expanded by body1
         | %ebp-32 |             |9   removed by finish1
         | %ebp-36 |             |10
%esp ->  | %ebp-40 | n-2         |11  "fibo(n)_4w"
         |---------+-------------|.............................
         |stack frame for fibo   |
```

F_n calling sequence in detail

fibo(n) call (n>2)

a	fibo(n) setup	
b	fibo(n) body1	call to fibo(n-2)
c	fibo(n) body2	call to fibo(n-1)
d	fibo(n) finish	

fibo(n) call (n<=2)

a	fibo(n) setup
b	fibo(n) finish

- skip over body1 body2

F_3 calling sequence

1.	fibo(3)	
2.		fibo(1)
3.		fibo(2)

F_3 calling sequence in detail

1.a 1.b	fibo(3) setup fibo(3) body1	
2.a 2.b		fibo(1) setup fibo(1) finish
1.c	fibo(3) body2	
3.a 3.b		fibo(2) setup fibo(2) finish
1.d	fibo(3) finish	

Stack frames for `fibo(3)` computation

fibo(3) stack frames

fush(3) setup	fibo(3) body1	fibo(1) setup	fibo(1) fini
fibo(3)_7w	fibo(3)_7w	fibo(3)_7w	fibo(3)_7w
	fibo(3)_4w1	fibo(3)_4w1	fibo(3)_4w1
		fibo(1)_7w	

fibo(3) body2	fibo(2) setup	fibo(2) fini	fibo(3) fini
fibo(3)_7w	fibo(3)_7w	fibo(3)_7w	
fibo(3)_4w1	fibo(3)_4w1	fibo(3)_4w1	
fibo(3) 4w2	fibo(3)_4w2	fibo(3)_4w2	
	fibo(2)_7w		

F_4 calling sequence

1.	fibo(4)		
2.		fibo(2)	
3.		fibo(3)	
4.			fibo(1)
5.			fibo(2)

F_4 calling sequence in detail

1.a 1.b	fibo(4) setup fibo(4) body1		
2.a 2.b		fibo(2) setup fibo(2) finish	
1.c	fibo(4) body2		
3.a 3.b		fibo(3) setup fibo(3) body1	
4.a 4.b			fibo(1) setup fibo(1) finish
3.c		fibo(3) body2	
5.a 5.b			fibo(2) setup fibo(2) finish
3.d		fibo(3) finish	
1.d	fibo(4) finish		

Stack frames for fibo(4) computation (a)

fibo(4) stack frames (a)

fibo(4) setup	fibo(4) body1	fibo(2) setup	fibo(2) fini
fibo(4)_7w	fibo(4)_7w	fibo(4)_7w	fibo(4)_7w
	fibo(4)_4w1	fibo(4)_4w1	fibo(4)_4w1
		fibo(2)_7w	

fibo(4) body2	fibo(3) setup	fibo(3) body1	fibo(1) setup
fibo(4)_7w	fibo(4)_7w	fibo(4)_7w	fibo(4)_7w
fibo(4)_4w1	fibo(4)_4w1	fibo(4)_4w1	fibo(4)_4w1
fibo(4)_4w2	fibo(4)_4w2	fibo(4)_4w2	fibo(4)_4w2
	fibo(3)_7w	fibo(3)_7w	fibo(3)_7w
		fibo(3)_4w1	fibo(3)_4w1
			fibo(1)_7w

Stack frames for fibo(4) computation (b)

fibo(4) stack frames (b)

fibo(1) fini	fibo(3) body2	fibo(2) setup	fibo(2) fini
fibo(4)_7w	fibo(4)_7w	fibo(4)_7w	fibo(4)_7w
fibo(4)_4w1	fibo(4)_4w1	fibo(4)_4w1	fibo(4)_4w1
fibo(4)_4w2	fibo(4)_4w2	fibo(4)_4w2	fibo(4)_4w2
fibo(3)_7w	fibo(3)_7w	fibo(3)_7w	fibo(3)_7w
fibo(3)_4w1	fibo(3)_4w1	fibo(3)_4w1	fibo(3)_4w1
	fibo(3)_4w2	fibo(3)_4w2	fibo(3)_4w2
		fibo(2)_7w	

fibo(3) fini	fibo(4) fini	. . . . .	. . . . .
fibo(4)_7w			
fibo(4)_4w1			
fibo(4)_4w2			