

Stack Frame Mechanism used by C Compiler for Procedure Calls

Subroutine structure

```
subr:
    pushl %ebp
    movl %esp,%ebp
    subl %(size_of_memory_for_local_variables),%esp

    ... body of subroutine ...

    movl %ebp,%esp
    popl %ebp
    ret
```

Example

```
void main() {          main:
    void swap();       pushl %ebp           ! save old bp
    int a,b;           movl %esp,%ebp      ! set new bp as current sp
    a = 5; b = 44;     subl $8,%esp        ! sub for a and b
    swap(&a,&b);        movl $5,-4(%ebp)    ! initialize a
}                      movl $44,-8(%ebp)   ! initialize b
                      leal -8(%ebp),%eax    ! form addr of b
                      pushl %eax         ! push onto stack
                      leal -4(%ebp),%eax  ! form addr of a
                      pushl %eax         ! push onto stack
                      call swap          ! call - save return PC on stack
                      movl %ebp, %esp     ! discard local variables from stack
                      popl %ebp          ! discard stack frame
                      ret

void swap(x,y)        swap:
    int *x,*y;        pushl %ebp           ! save old bp
    {                 movl %esp,%ebp      ! set new bp as current sp
        int temp;     subl $4,%esp         ! sub for temp
        temp = *x;    movl 8(%ebp),%eax   ! move addr x into eax
        *x = *y;      movl (%eax),%edx    ! indirectly get value in a
        *y = temp;    movl %edx,-4(%ebp)   ! store into temp
        return;       movl 8(%ebp),%eax   ! move addr x into eax
    }                 movl 12(%ebp),%edx   ! move addr y into edx
                      movl (%edx),%ecx   ! indirectly get value in b
                      movl %ecx,(%eax)   ! store indirectly into a
                      movl 12(%ebp),%eax ! move addr y into eax
                      movl -4(%ebp),%edx ! move temp into edx
                      movl %edx,(%eax)   ! store indirectly into b
                      movl %ebp,%esp     ! discard local variables from stack
                      pop %ebp           ! discard stack frame
                      ret                ! load PC from stack
```

For more information see IA-32 Intel Architecture Software Developer's Manual Volume 1: Basic Architecture, Chapter 6, "Procedure Calls, Interrupts, and Exceptions" (<http://developer.intel.com/design/processor/manuals/253665.pdf>)