

# Stack Frame

Note Title

10/22/2007

Pass parameters

Caller pushes args on stack, last to first

```
foo ( int a, int b)
```

```
} ... a + b ...
```

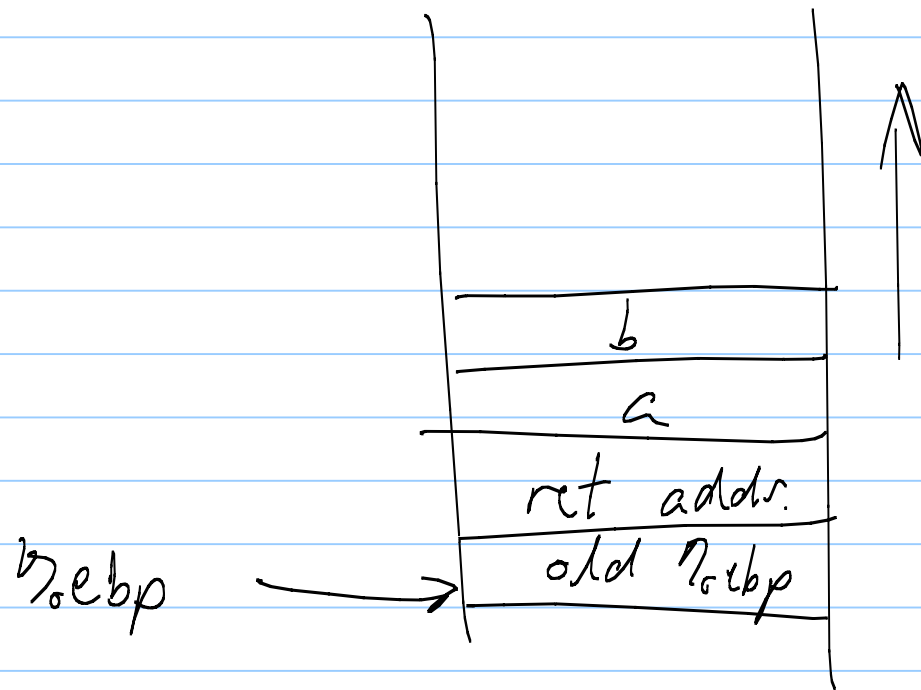
main:

```
...  
pushl b  
pushl a  
call foo  
addl $8, %esp // deallocate args  
...
```

parameters can be referenced using frame pointer

foo:

```
...  
movl 8(%ebp), %edx // edx ← a  
addl 12(%ebp), %edx // edx ← a + b
```



Maintaining stack frame is callee's job

proc:      pushl      %ebp      // adjust frame

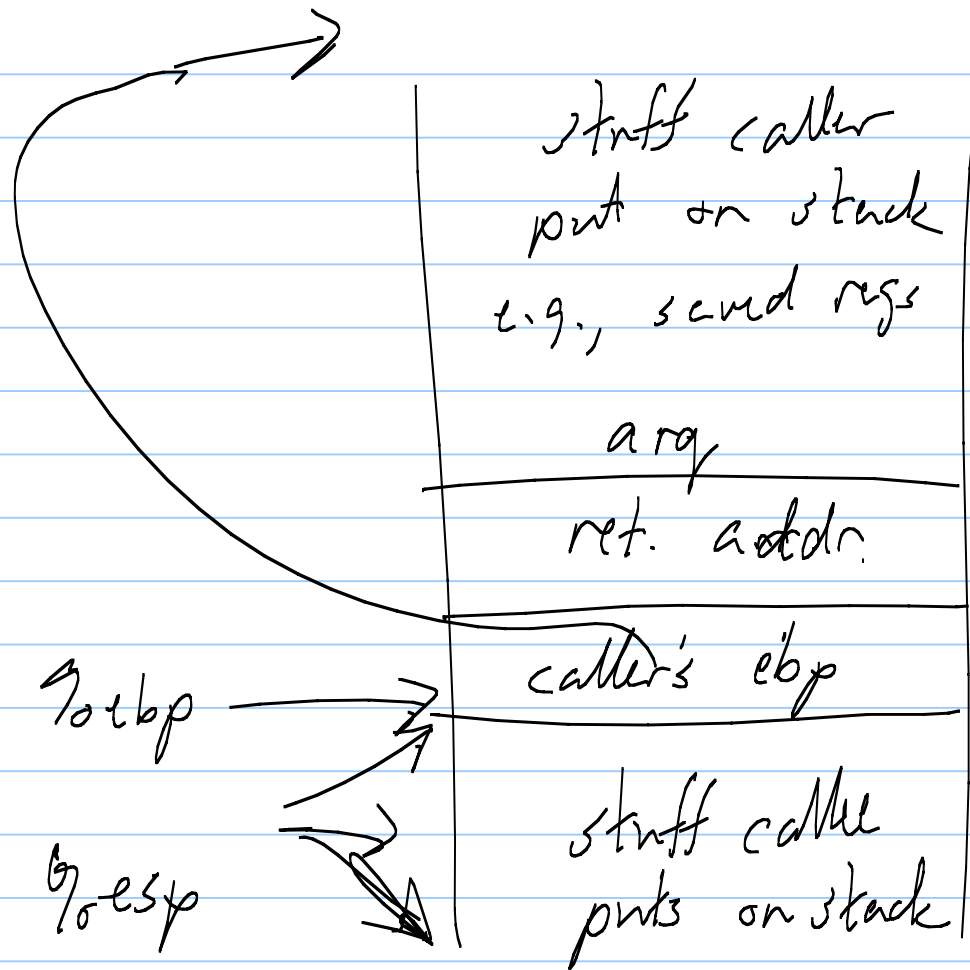
          movl      %esp, %ebp

          :

          movl      %ebp, %esp      // restore frame

          popl

          ret



Local variables

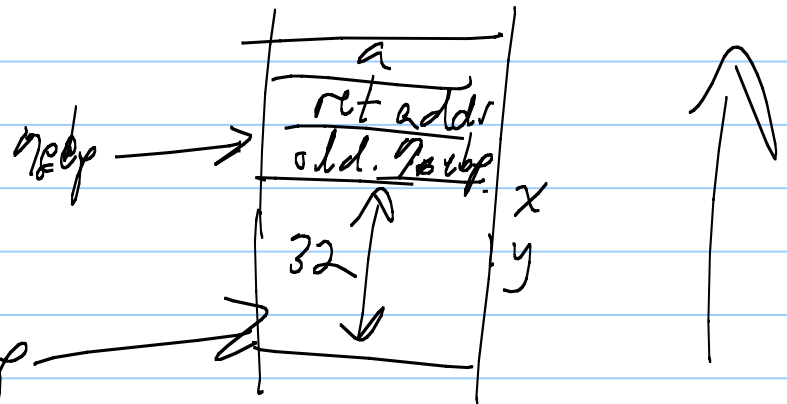
caller allocates stack space for locals

```
bar (int a)
    int x, y, ...
```

```
bar:  pushl %ebp //adjust frame
      movl %esp, %ebp
      subl $32, %esp
```

```
    x = a + y;
```

```
movl -8(%ebp), %eax
addl 8(%ebp), %eax
movl %eax, -4(%ebp)
```



caller

save caller regs  
push args  
call routine  
deallocate args  
restore regs

callee

adjust stack frame  
allocate local vars  
save caller regs  
:  
:  
restore regs  
deallocate local vars  
restore frame  
return

return val in %eax if it fits,  
if 24 bytes %eax holds ptr value

```
int func (int i) {  
    return i;  
}
```

```
func:  
    pushl   %ebp  
    movl   %esp, %ebp  
    movl   8(%ebp), %eax  
    movl   %eax, %esp  
    popl   %ebp  
    ret
```

<pre> int add(int x, int y) {     int sum;      sum = x + y;     return (sum); } </pre>	<pre> add: pushl %ebp //adj fr       movl %esp, %ebp       subl \$4, %esp //alloc sum       pushl %ebx //save reg       movl 8(%ebp), %ebx //x       addl 12(%ebp), %ebx //y       movl %ebx, -4(%ebp)       movl %eax, %eax //ret       popl %ebx       movl %ebp, %esp       popl %ebp       ret </pre>
---	---

$m = \text{add}(a, b)$

```
pushl %ecx // save reg
pushl %edx // save reg
pushl b // push args
pushl a // push args
call add // call
addl $8, %esp // deallocate
popl %edx // restore reg
popl %ecx // restore reg
movl %eax, m // m ← add(a, b)
```

gets (string)

char string [N]

input string until \n w/ no  
bounds check?