

Stack Frame

Note Title

10/22/2007

Pass parameters

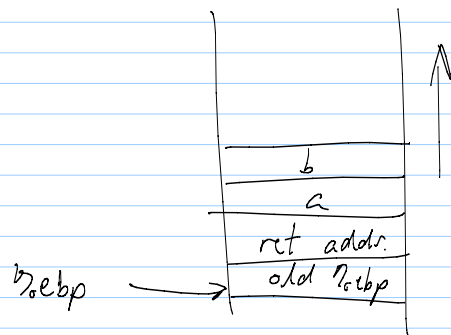
Caller pushes args on stack, last to first

```
foo (int a, int b)
{ ... a+b ...
}
```

```
main:
...
pushl b
pushl a
call foo
addl $8, %esp //deduct args
...
```

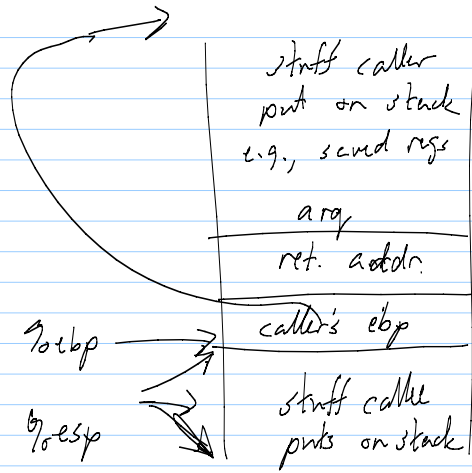
parameters can be referenced using frame pointer

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



Maintaining stack frame is caller's job

```
proc:
pushl %ebp //adjust frame
movl %esp, %ebp
:
movl %ebp, %esp //restore frame
popl %ebp
ret
```



Local variables

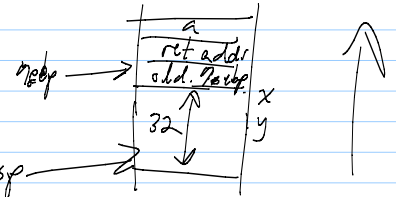
caller allocates stack space for locals

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

```
bar: pushl %ebp //adjust
      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 32 bytes $\%eax$ holds ptr value

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

```
func: pushl  %ebp
      movl  %esp, %ebp
      movl  8(%ebp), %eax
      movl  %eax, -4(%esp)
      popl  %ebp
      retl
```

```

int add(int x, int y) add: pushl %ebp //adjust
int sum; movl %esp, %ebp
sum = x+y; subl $4, %esp //alloc sum
return (sum); pushl %ebx //save reg
3 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

```

m = add(a, b)

```

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

```

gets (string)
char string[N]
input string until \r w/ no
bounds check?