Saving registers

```
pushl %eax  } caller
pushl %edx  } saved
call  func
```

```
func:
    push %ebx  } callee
    push %esi  } saved
```

```
func:
    subl  $24, %esp
    movl  %ebx, 4(%esp)
```

---

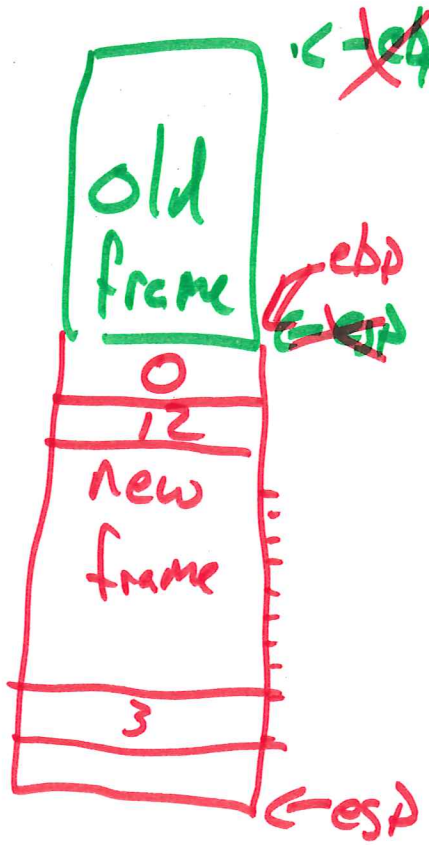
local variables

```
int some_func() {
    int i = 0;      -> 4
    int j = 12;     -> 4
    int arr[10] = {1, ..., 3} -> 40+4
```

=> 52 bytes

```
some_func:
        pushl %ebp
        movl  %esp, %ebp
        subl  $52, %esp
        movl  $0,-4(%ebp)
        movl  $12,+48(%esp)
        movl  $12,-8(%ebp)
        movl
              ;
              ;
```

the same



Things to do before calling

1) Save caller-saved registers
2) push parameters onto stack
3) call inst.

Things to do when called

1) Save old ebp
2) Set ebp to current frame base (esp)

2) Allocate + initialize local vars.

3) Save callee-saved registers (if they are used)

Things to do before return

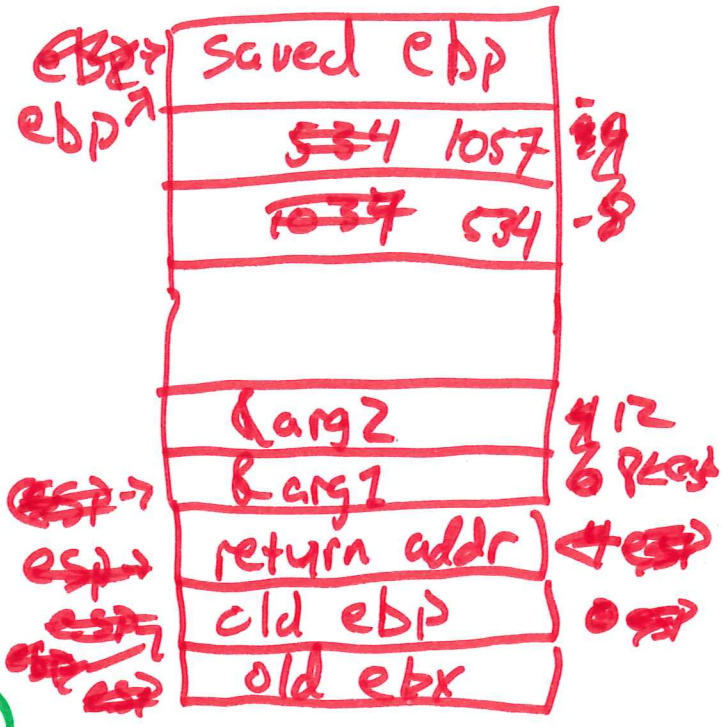
1) restore callee-saved registers
2) set the return value
3) fix up the stack (leave) ← maybe
4) return

Example → Book section 3.7.4

```
int swap_add(int *xp, int *yp) {
    int x = *xp;
    int y = *yp;

    *xp = y;
    *yp = x;
    return x+y;
}

int caller() {
    int arg1 = 534;
    int arg2 = 1057;

    int sum = swap_add(&arg1, &arg2)
    int diff = arg1 - arg2;

    return sum * diff;
}
```

caller:

```
pushl %ebp
movl %esp, %ebp
subl $24, %esp
movl $534, -4(%ebp)
movl $1057, -8(%ebp)
leal -8(%ebp), %eax
movl %eax, 4(%esp)
leal -4(%ebp), %ecx
movl %eax, (%esp)
call swap_add

movl -4(%ebp), %edx
subl
leal -8(%ebp), %edx
imull %edx, %eax
leave
ret
```



```
swap_add:
        pushl %ebp
        movl %esp, %ebp
        ~~movl~~ *
        pushl %ebx
        movl 8(%ebp), %edx
        movl 12(%ebp), %ecx
        movl (%edx), %ebx
        movl (%ecx), %eax
        movl %ebx, (%ecx)
        movl %eax, (%edx)
        addl %ebx, %eax
        popl %ebx
        popl %ebp
        ret
```