Saving registers
pushl %eax 3 caller
pushl %edx saved
call func

func:
push %ebx 3 callee
push %esi 3 saved

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

local variables

int some_func() 52 bytes
int i = 0; 74
int j = 42; 74
int arr[10] = 3... 3 -> 40+4
Some function:
- pushl %ebp
- movl %esp, %ebp
- subl $52, %esp
- movl $0, -4(%ebp)
- movl $12, -8(%ebp)
  
  Things to do before calling:
  1) Save caller-saved registers
  2) Push parameters onto stack
  3) Call instruction

  Things to do when called:
  1) Save old ebp
  2) Set ebp to current frame base (esp)
4) 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 (maybe)
3) Fix up the stack (leave)
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;
}
```
```assembly
; caller:
pushl %ebp
movl %esp, %ebp
subl $24, %esp
movl $584, -4(%ebp)
movl $057, -8(%ebp)
leave -8(%ebp), %eax
movl %eax, 4(%esp)
leave -4(%ebp), %ecx
movl %eax, (%esp)
call swap_add
movl -4(%ebp), %edx
subl -8(%ebp), %edx
imull %edx, %eax
leave
return
```
Swap add:

```assembly
pushl %ebp
movl %esp, %ebp
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
```