x86 Call/Return Protocol

1. Save "caller-save" registers (%eax, %ecx, %edx) onto stack (opt)

```
push %ecx
```

2. Push arguments onto stack, in reverse order (argN,…, arg1)

```
push 0x10 // argument 2
push 0x20 // argument 1
```

3. Call function (which pushes return address onto stack)

```
call 0x80400000
```

4. Establish new base pointer (saving old one)

```
push %ebp
movl %esp, %ebp
```

5. Allocation: Make room for local variables on stack

```
subl $8, %esp
```

6. Save "callee-save" registers (%ebx, %esi, %edi) onto stack

```
push %ebx
```

7. Execute body of routine (use ebp to access args, locals);

```
 8(%ebp)   // argument 1
12(%ebp)   // argument 2
-4(%ebp)   // local var
```

8. Restore "callee-save" registers

```
pop %ebx
```

9. Deallocation: Free local stack space

```
movl %ebp, %esp
```

10. Restore old base pointer

```
pop %ebp
```

11. Return from function (popping return address off of stack)

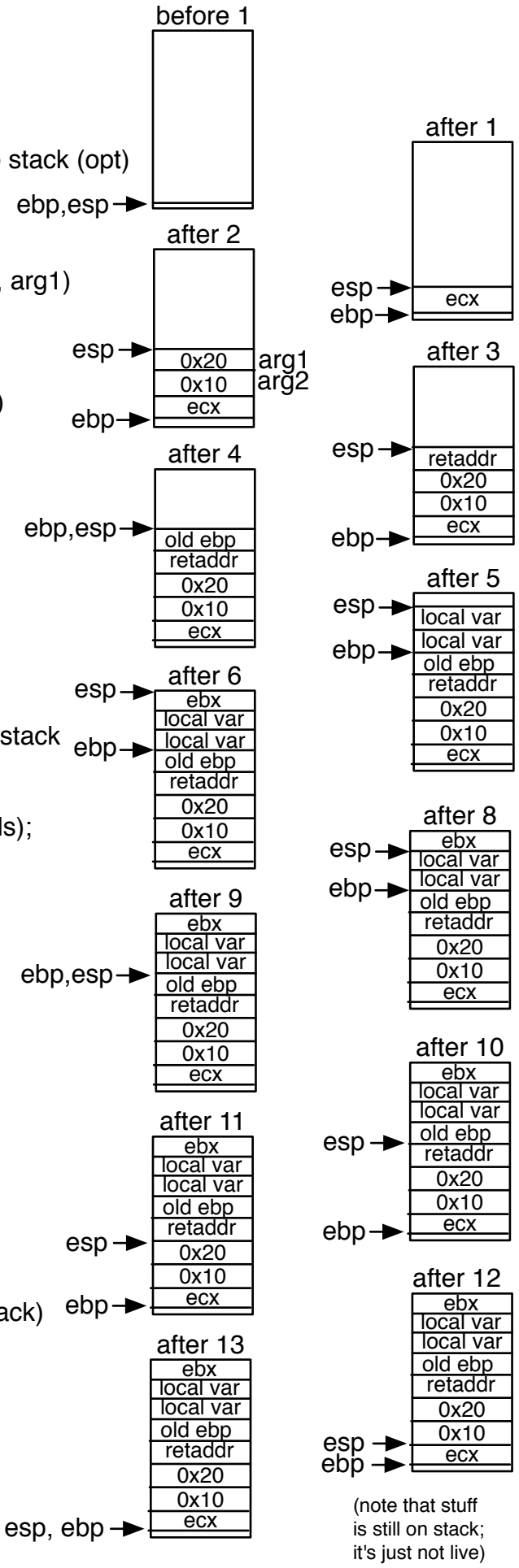
```
ret
```

12. Deallocate params

```
addl $8, %esp
```

13. Restore "caller-save" registers

```
pop %ecx
```



(note that stuff is still on stack; it's just not live)