Recursive Example

factorial

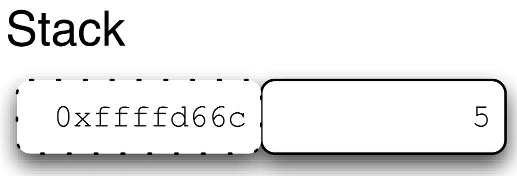
```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

just before the call to factorial

_start
```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```



just after the call to factorial(5)

factorial

```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

Stack

0xffffd664	0x00000000	old EBP
0xffffd668	0x0804805b	line in _start after call to factorial
0xffffd66c	5	

about to call factorial(4)

factorial

```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

Stack

0xffffd660	4	
0xffffd664	0x00000000	old EBP
0xffffd668	0x0804805b	line in _start after call to factorial
0xffffd66c	5	

about to call factorial(3)

factorial

```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

Stack

0xffffd654	3	
0xffffd658	0xffffd664	old EBP
0xffffd65c	0x08048079	line in factorial after call to factorial
0xffffd660	4	
0xffffd664	0x00000000	old EBP
0xffffd668	0x0804805b	line in _start after call to factorial
0xffffd66c	5	

about to call factorial(2)

factorial

```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

Stack

0xffffd648	2	
0xffffd64c	0xffffd658	old EBP
0xffffd650	0x08048079	line in factorial after call to factorial
0xffffd654	3	
0xffffd658	0xffffd664	old EBP
0xffffd65c	0x08048079	line in factorial after call to factorial
0xffffd660	4	
0xffffd664	0x00000000	old EBP
0xffffd668	0x0804805b	line in _start after call to factorial
0xffffd66c	5	

about to call factorial(1)

factorial

```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

Stack

0xffffd63c	1	
0xffffd640	0xffffd64c	old EBP
0xffffd644	0x08048079	line in factorial after call to factorial
0xffffd648	2	
0xffffd64c	0xffffd658	old EBP
0xffffd650	0x08048079	line in factorial after call to factorial
0xffffd654	3	
0xffffd658	0xffffd664	old EBP
0xffffd65c	0x08048079	line in factorial after call to factorial
0xffffd660	4	
0xffffd664	0x00000000	old EBP
0xffffd668	0x0804805b	line in _start after call to factorial
0xffffd66c	5	

in factorial(1)

factorial

```
0x08048067 <+0>:     push   %ebp
0x08048068 <+1>:     mov    %esp,%ebp
0x0804806a <+3>:     mov    0x8(%ebp),%eax
0x0804806d <+6>:     cmp    $0x1,%eax
0x08048070 <+9>:     je     0x804807f <end_factorial>
0x08048072 <+11>:    dec    %eax
0x08048073 <+12>:    push   %eax
0x08048074 <+13>:    call   0x8048067 <factorial>
0x08048079 <+18>:    mov    0x8(%ebp),%ebx
0x0804807c <+21>:    imul   %ebx,%eax
```

_start

```
0x08048054 <+0>:     push   $0x5
0x08048056 <+2>:     call   0x8048067 <factorial>
0x0804805b <+7>:     add    $0x4,%esp
0x0804805e <+10>:    mov    %eax,%ebx
0x08048060 <+12>:    mov    $0x1,%eax
0x08048065 <+17>:    int    $0x80
```

Stack

0xffffd634	0xffffd640	old EBP
0xffffd638	0x08048079	line in factorial after call to factorial
0xffffd63c	1	
0xffffd640	0xffffd64c	old EBP
0xffffd644	0x08048079	line in factorial after call to factorial
0xffffd648	2	
0xffffd64c	0xffffd658	old EBP
0xffffd650	0x08048079	line in factorial after call to factorial
0xffffd654	3	
0xffffd658	0xffffd664	old EBP
0xffffd65c	0x08048079	line in factorial after call to factorial
0xffffd660	4	
0xffffd664	0x00000000	old EBP
0xffffd668	0x0804805b	line in _start after call to factorial
0xffffd66c	5	

How did we get these values? GDB.

- to get the code in assembly:
  - `disas` *`label`* ***e.g.,***
    - `disas _start`
    - `disas factorial`
- to read the value stored in register %ebp
  - `p $ebp`
- to read a series of 20 values stored on the stack starting with the value stored in %esp:
  - `x/20x $esp`