Virtual Memory. The Stack

Adrian Ștendroiu
(sendroiu.adrian@gmail.com)

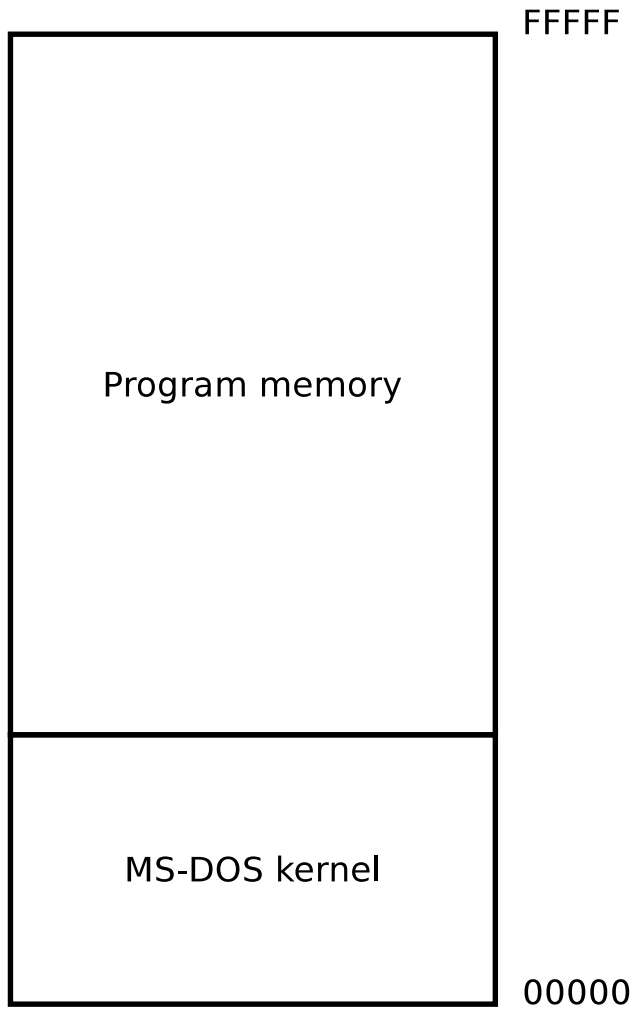
9 July 2013

ROSEdu Summer Workshops

Overview

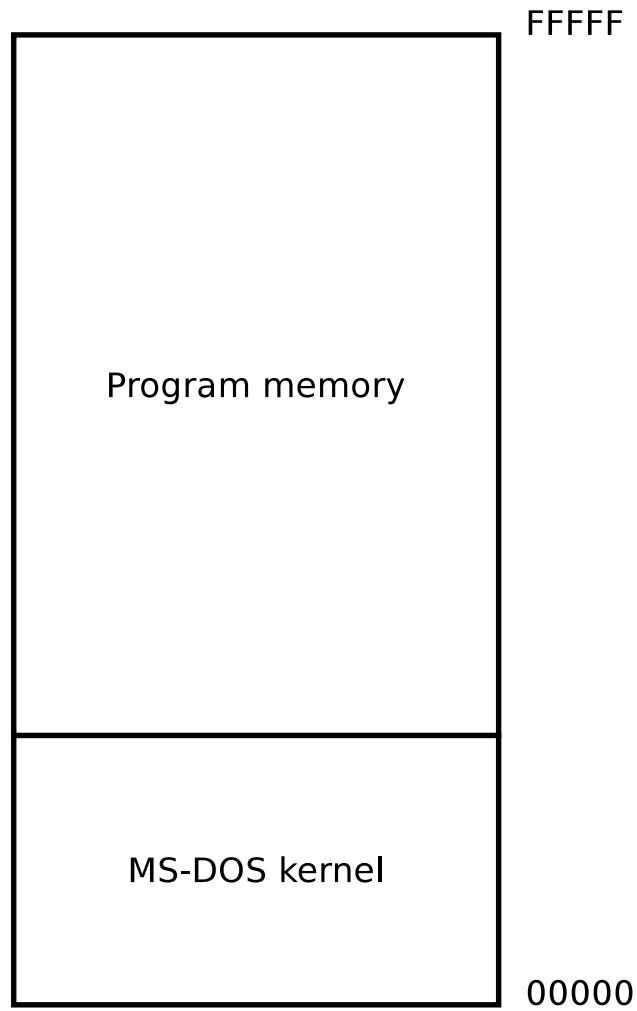
1. Virtual Memory
2. The Stack

8086 Days



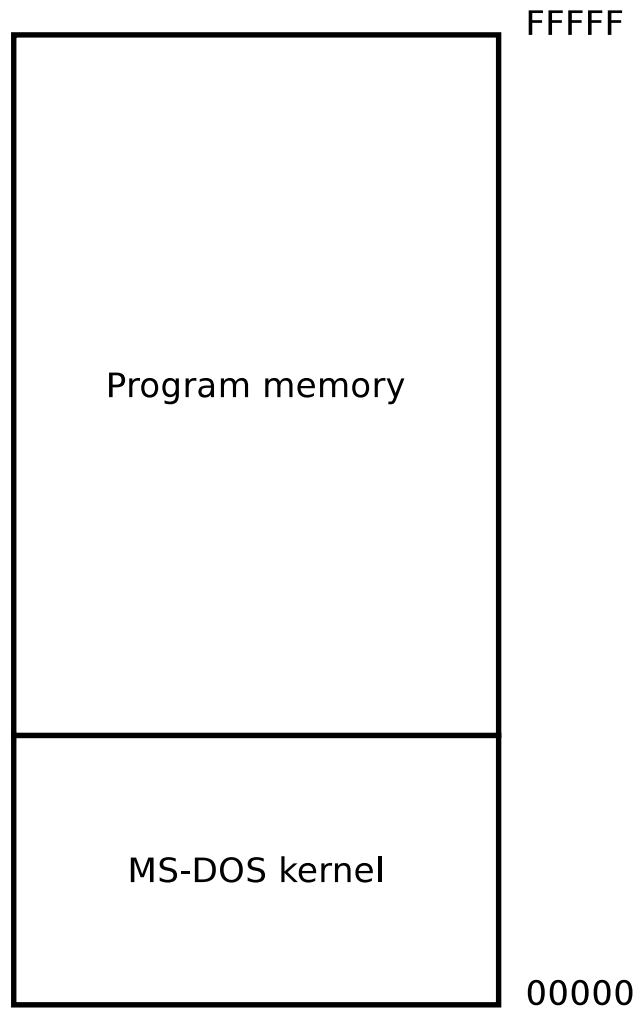
8086 Days

- A program can access the entire memory



8086 Days

- A program can access the entire memory
- A program can access the hardware directly



Virtual Memory

```c
#include <stdio.h>

int main()
{
        int x;

        printf("%x\n", (unsigned)&x);

        return 0;
}
```

Virtual Memory

```c
#include <stdio.h>

int main()
{
        int x;

        printf("%x\n", (unsigned)&x);

        return 0;
}
```

```
root@kali:/tmp# ./a.out
bffff77c
```

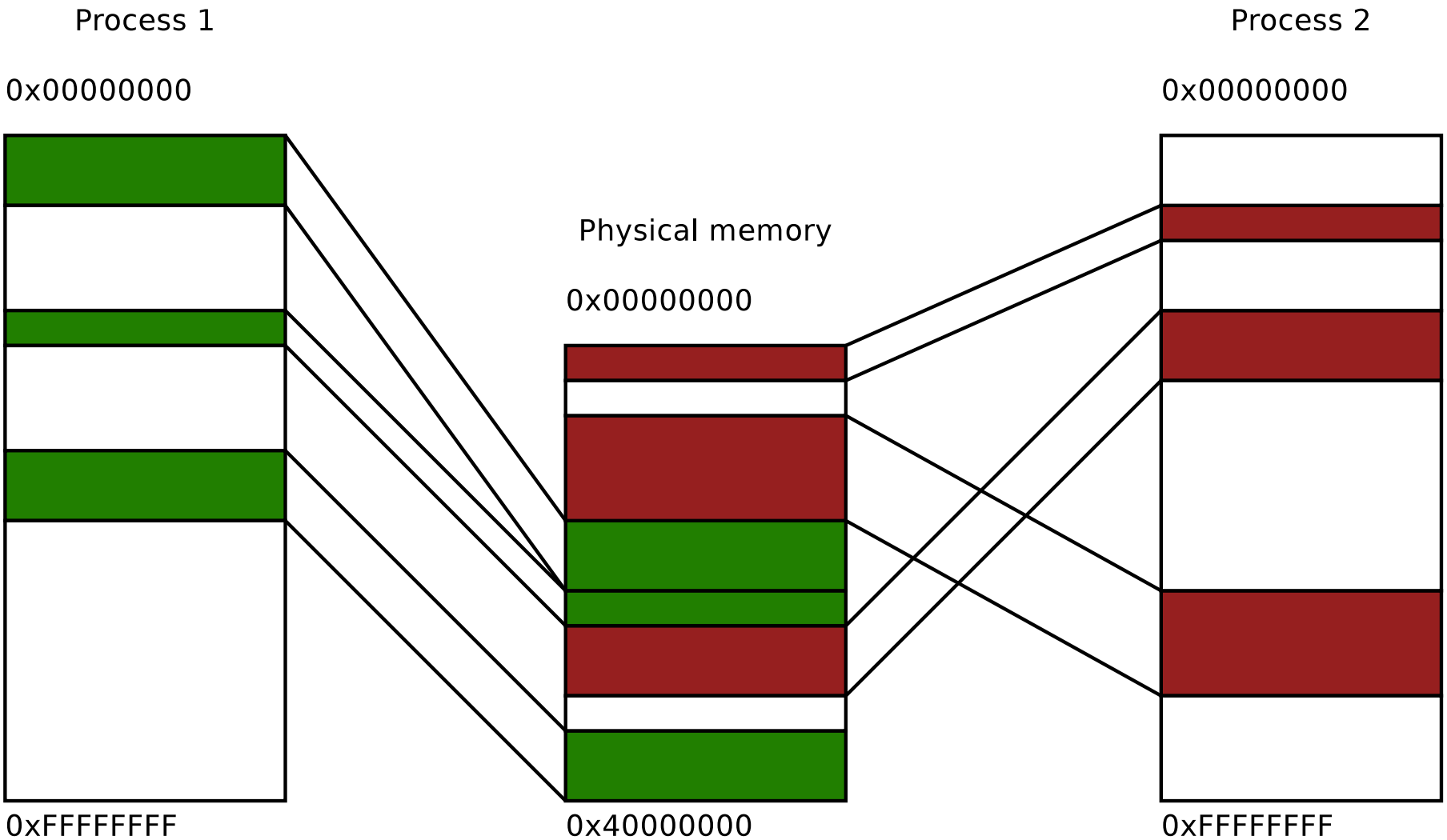
Virtual Memory

```c
#include <stdio.h>

int main()
{
        int x;

        printf("%x\n", (unsigned)&x);

        return 0;
}
```

```
root@kali:/tmp# ./a.out
bffff77c
```

```
root@kali:/tmp# ./a.out & ./a.out
[2] 17296
bffff77c
bffff77c
```

Virtual Memory



Virtual Memory

- Advantages

Virtual Memory

- Advantages
  - Isolation

Virtual Memory

- Advantages
  - Isolation
  - Easy to program

Virtual Memory

- Advantages
  - Isolation
  - Easy to program
  - Additional useful mechanisms can be implemented

Virtual Memory

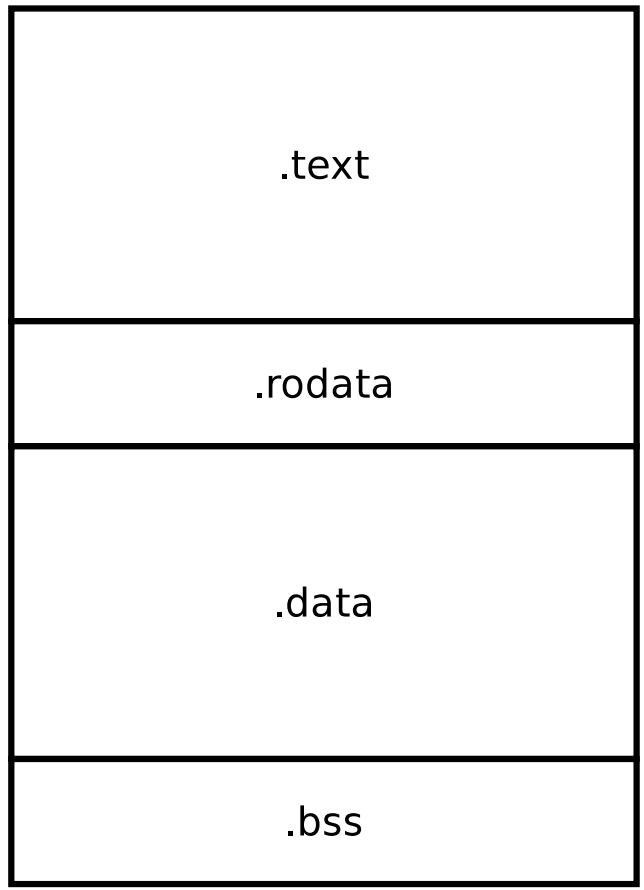
- Advantages
  - Isolation
  - Easy to program
  - Additional useful mechanisms can be implemented
    - Swapping

Virtual Memory

- Advantages
  - Isolation
  - Easy to program
  - Additional useful mechanisms can be implemented
    - Swapping
    - Shared memory

ELF

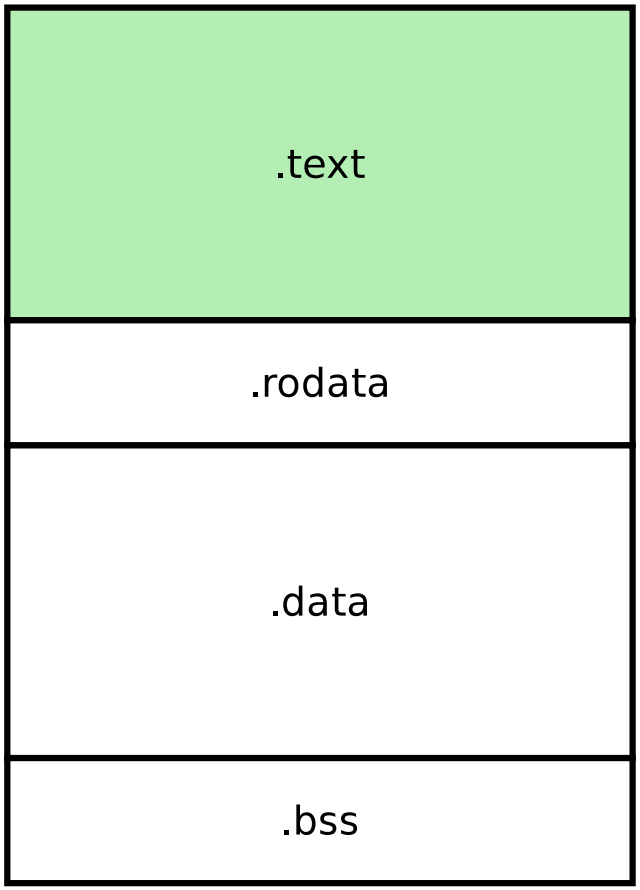
- Executable and Linkable Format
- Divided in sections



ELF

- .text - program code

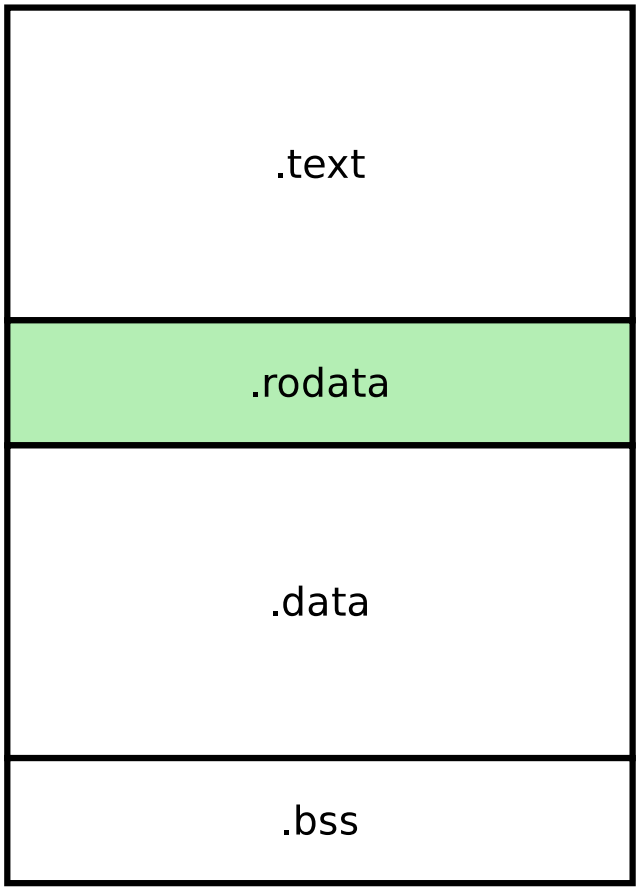
```c
#include <stdio.h>

const int x = 3;

int y = 10;
int v[] = { 0, 1, 2, 3 };

int z;
int w[100];

int f()
{
        return 0;
}

int main()
{
        printf("Hello, World\n");

        return 0;
}
```



ELF

- .rodata - global constant data

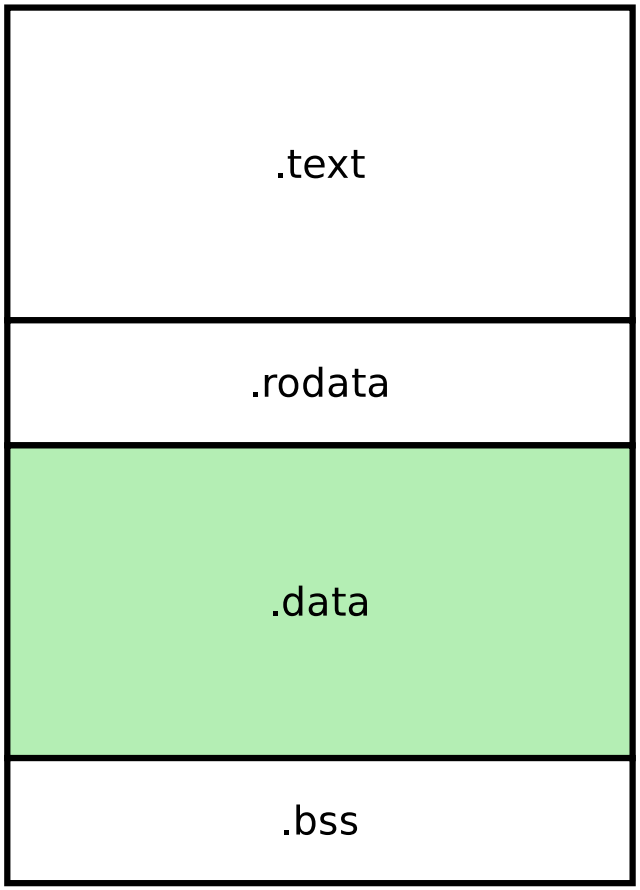
```c
#include <stdio.h>

const int x = 3;

int y = 10;
int v[] = { 0, 1, 2, 3 };

int z;
int w[100];

int f()
{
        return 0;
}

int main()
{
        printf("Hello, World\n");

        return 0;
}
```



ELF

- .data - global initialized data

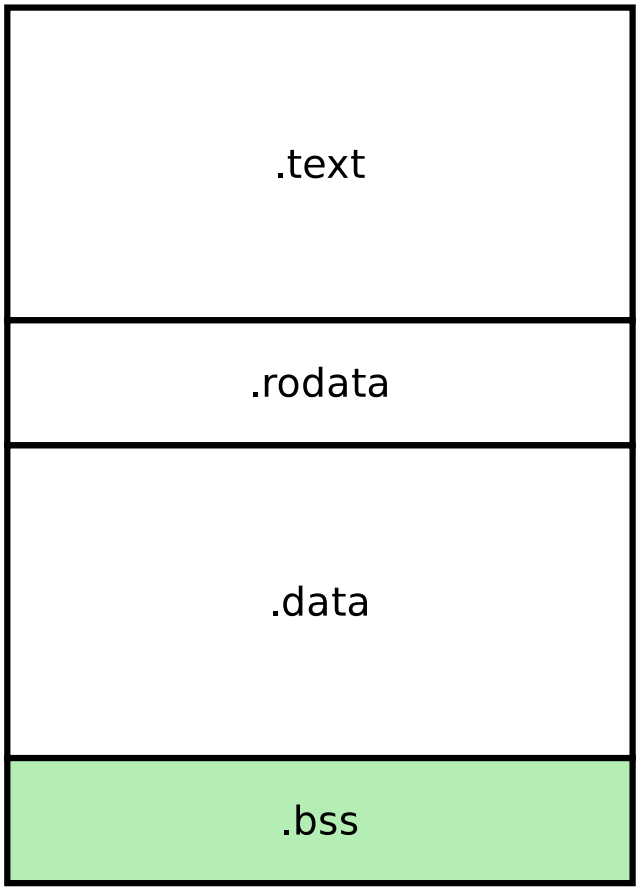
```c
#include <stdio.h>

const int x = 3;

int y = 10;
int v[] = { 0, 1, 2, 3 };

int z;
int w[100];

int f()
{
        return 0;
}

int main()
{
        printf("Hello, World\n");

        return 0;
}
```



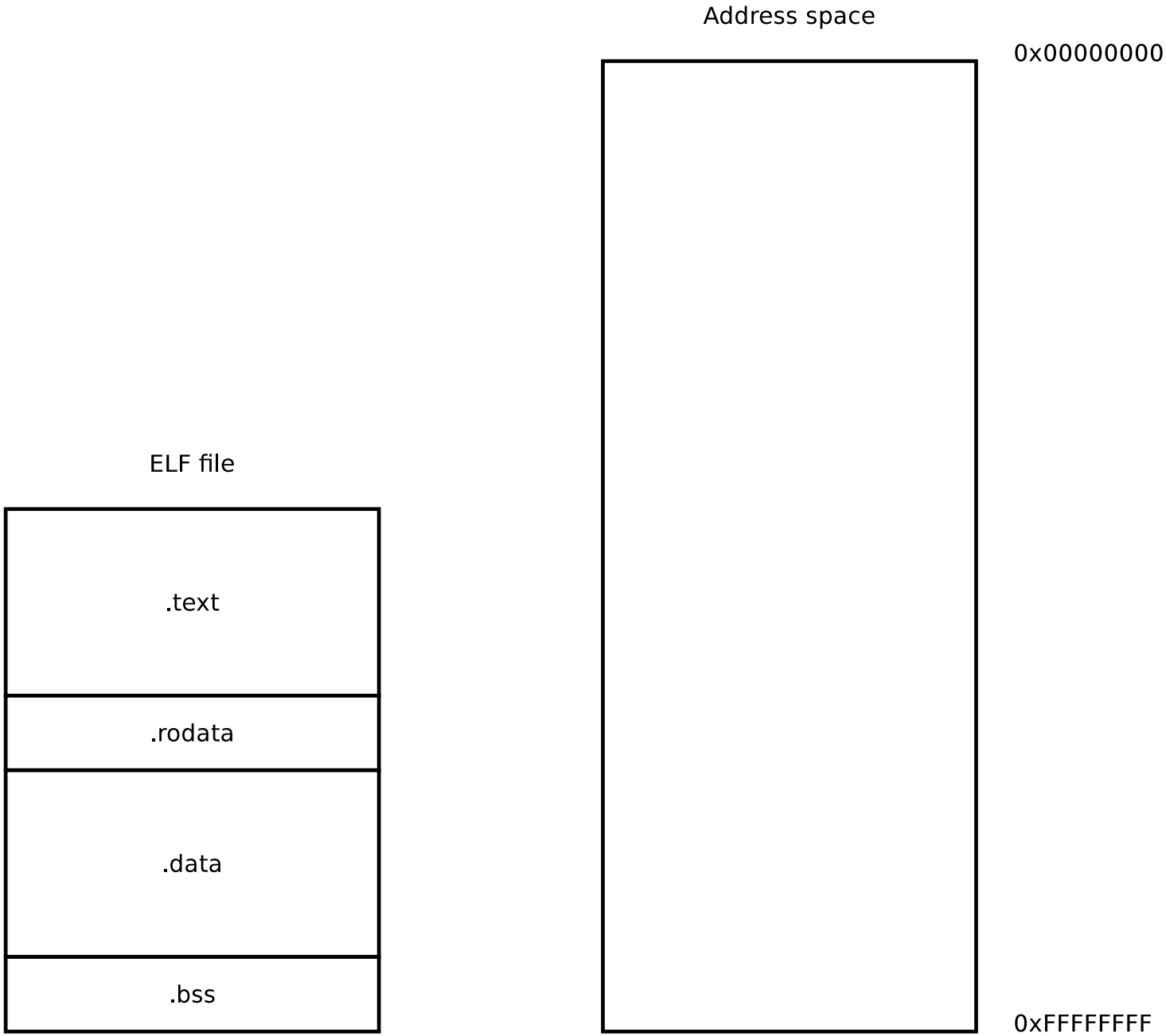
ELF

- .bss - global uninitialized data

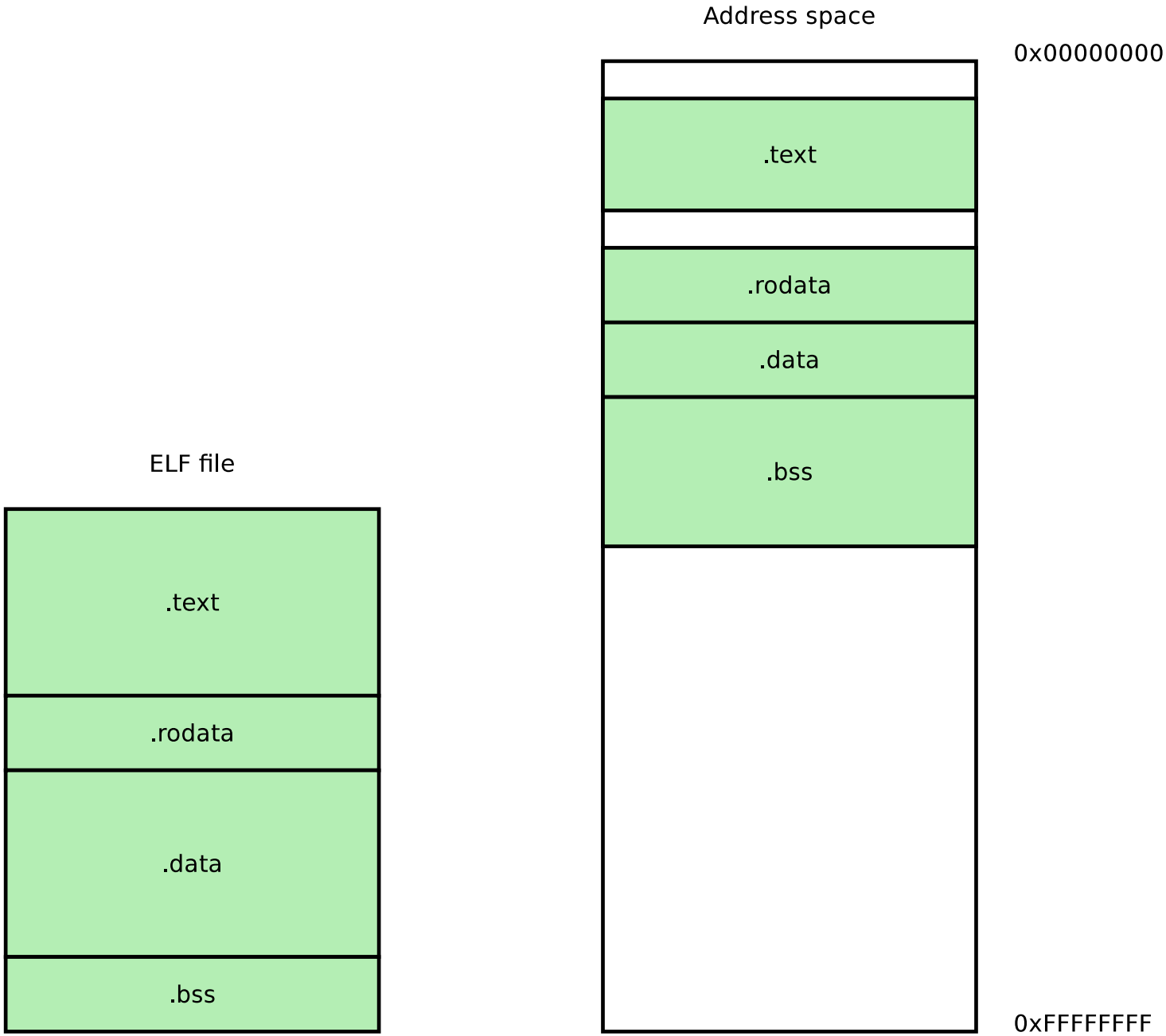
```c
#include <stdio.h>

const int x = 3;

int y = 10;
int v[] = { 0, 1, 2, 3 };

int z;
int w[100];

int f()
{
        return 0;
}

int main()
{
        printf("Hello, World\n");

        return 0;
}
```



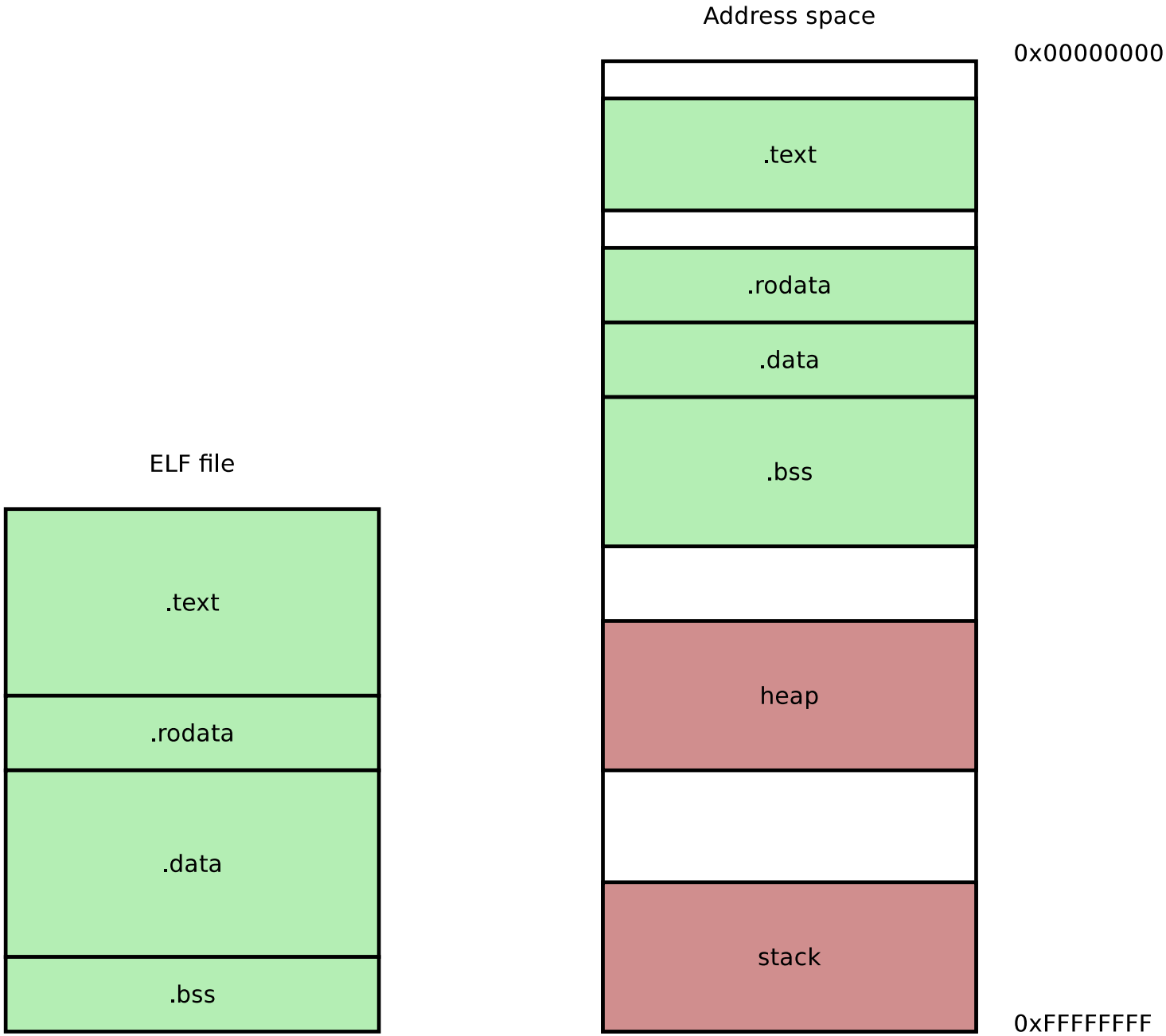
ELF Loading


Address space
0x00000000
0xFFFFFFFF
ELF file
.text
.rodata
.data
.bss

ELF Loading


Address space
0x00000000
.text
.rodata
.data
.bss
0xFFFFFFFF
ELF file
.text
.rodata
.data
.bss

ELF Loading


Address space
0x00000000
.text
.rodata
.data
.bss
heap
stack
0xFFFFFFFF
ELF file
.text
.rodata
.data
.bss

Heap and Stack

- Heap

Heap and Stack

- Heap
  - Handles dynamic memory allocations (malloc)

Heap and Stack

- Heap
  - Handles dynamic memory allocations (malloc)
- Stack

Heap and Stack

- Heap
  - Handles dynamic memory allocations (malloc)
- Stack
  - Local variables

Heap and Stack

- Heap
  - Handles dynamic memory allocations (malloc)
- Stack
  - Local variables
  - Temporary results

Heap and Stack

- Heap
  - Handles dynamic memory allocations (malloc)
- Stack
  - Local variables
  - Temporary results
  - Parameter passing

Heap and Stack

- Heap
  - Handles dynamic memory allocations (malloc)
- Stack
  - Local variables
  - Temporary results
  - Parameter passing
  - Storing return addresses

Overview

1 Virtual Memory

2 The Stack

The Stack

- A dedicated CPU register holds the current stack pointer
  - `esp` on i386
  - `rsp` on x86_64
  - `sp` on ARM

The Stack

- A dedicated CPU register holds the current stack pointer
  - `esp` on i386
  - `rsp` on x86_64
  - `sp` on ARM
- Grows downwards on most architectures
  - When pushing a value the stack pointer decreases

Calling Conventions

- A contract between the caller and the callee

Calling Conventions

- A contract between the caller and the callee
  - How parameters are passed

Calling Conventions

- A contract between the caller and the callee
  - How parameters are passed
  - What registers are caller saved, what registers are callee saved

Calling Conventions

- A contract between the caller and the callee
  - How parameters are passed
  - What registers are caller saved, what registers are callee saved
  - Who cleans the stack after the callee ends

Example: i386 architecture

- `cdecl` - Linux

Example: i386 architecture

- `cdecl` - Linux
  - Parameters passed on the stack, pushed in reverse order

Example: i386 architecture

- `cdecl` - Linux
  - Parameters passed on the stack, pushed in reverse order
  - EAX, ECX, EDX caller saved, rest callee-saved

Example: i386 architecture

- `cdecl` - Linux
  - Parameters passed on the stack, pushed in reverse order
  - EAX, ECX, EDX caller saved, rest callee-saved
  - Result in EAX

Example: i386 architecture

- `cdecl` - Linux
  - Parameters passed on the stack, pushed in reverse order
  - EAX, ECX, EDX caller saved, rest callee-saved
  - Result in EAX
  - The caller cleans the parameters from the stack

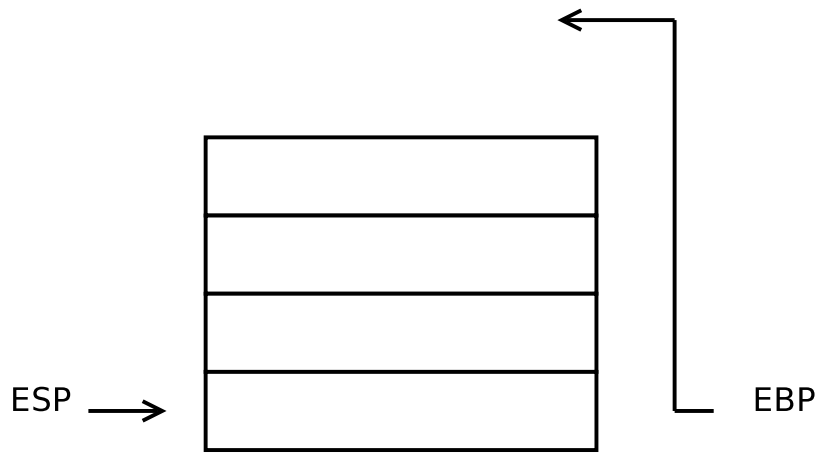
Example: i386 architecture

- `cdecl` - Linux
  - Parameters passed on the stack, pushed in reverse order
  - EAX, ECX, EDX caller saved, rest callee-saved
  - Result in EAX
  - The caller cleans the parameters from the stack
- `stdcall` - Win32

Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

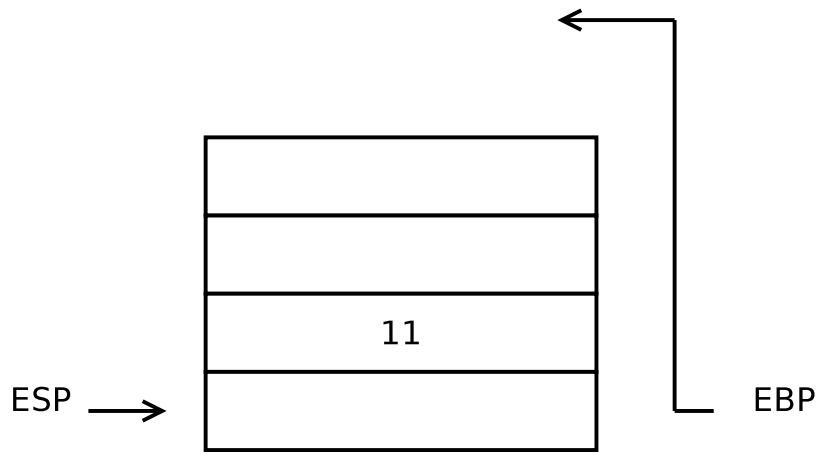
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

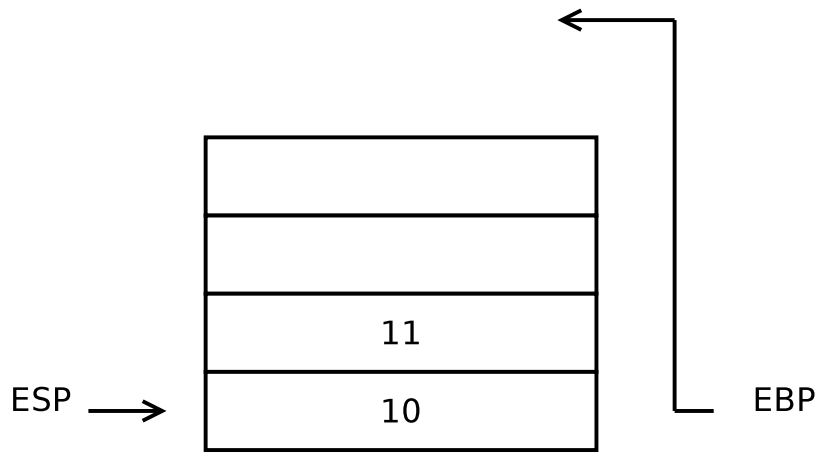
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

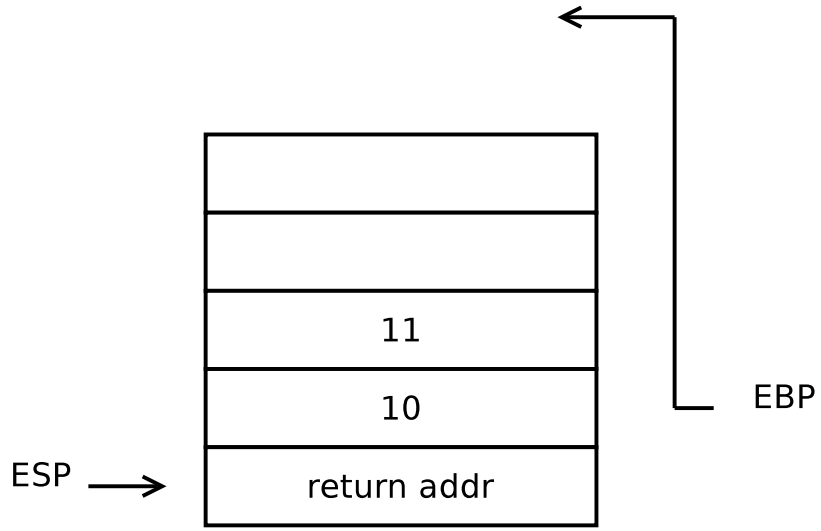
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

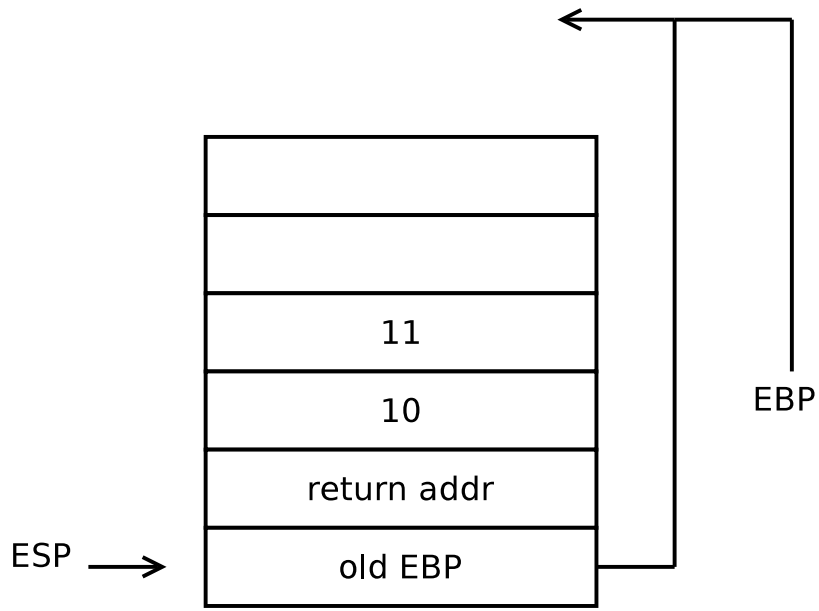
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

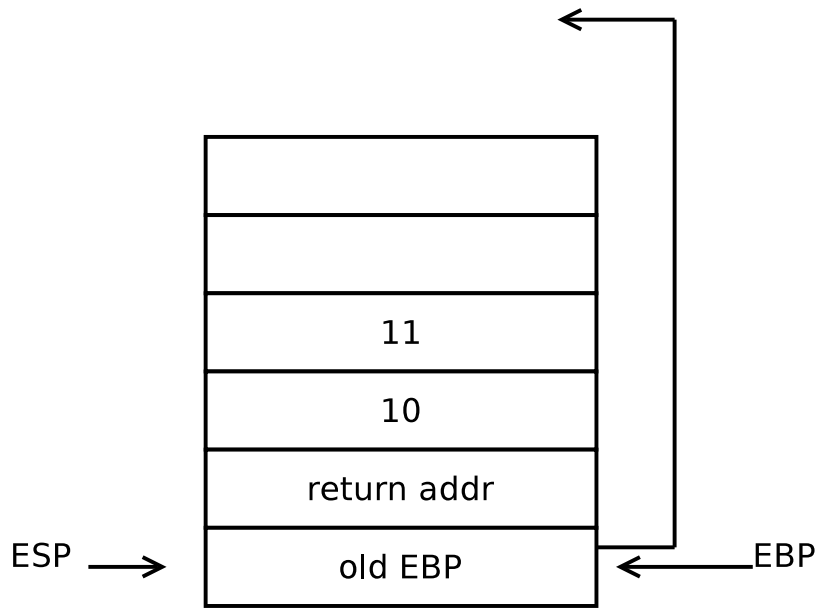
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

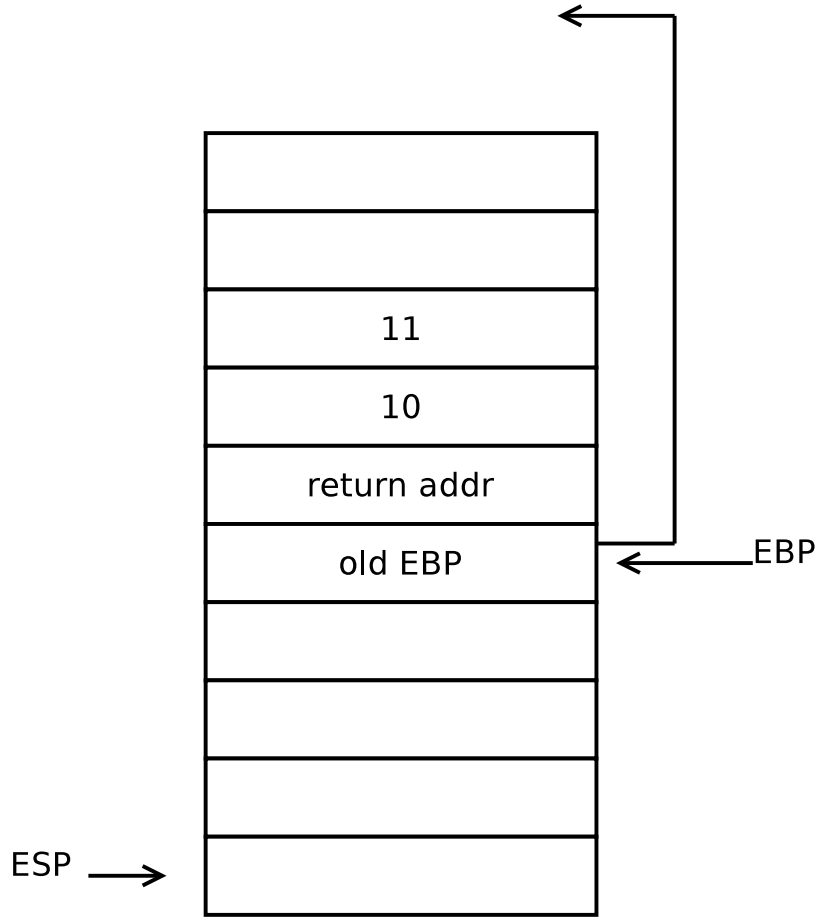
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

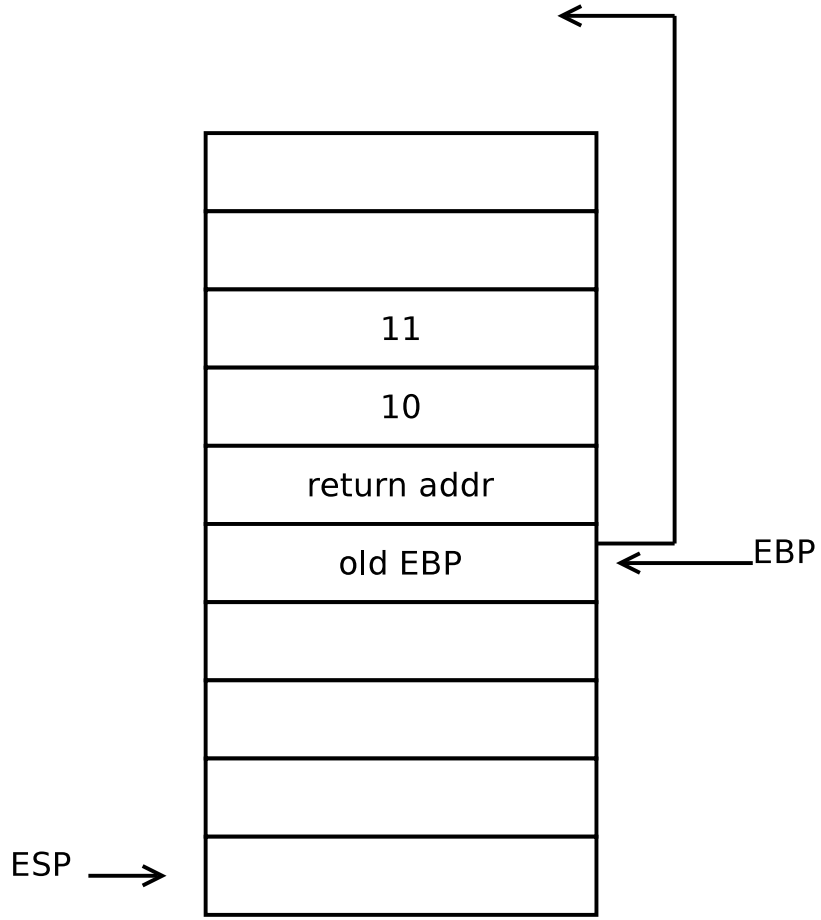
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

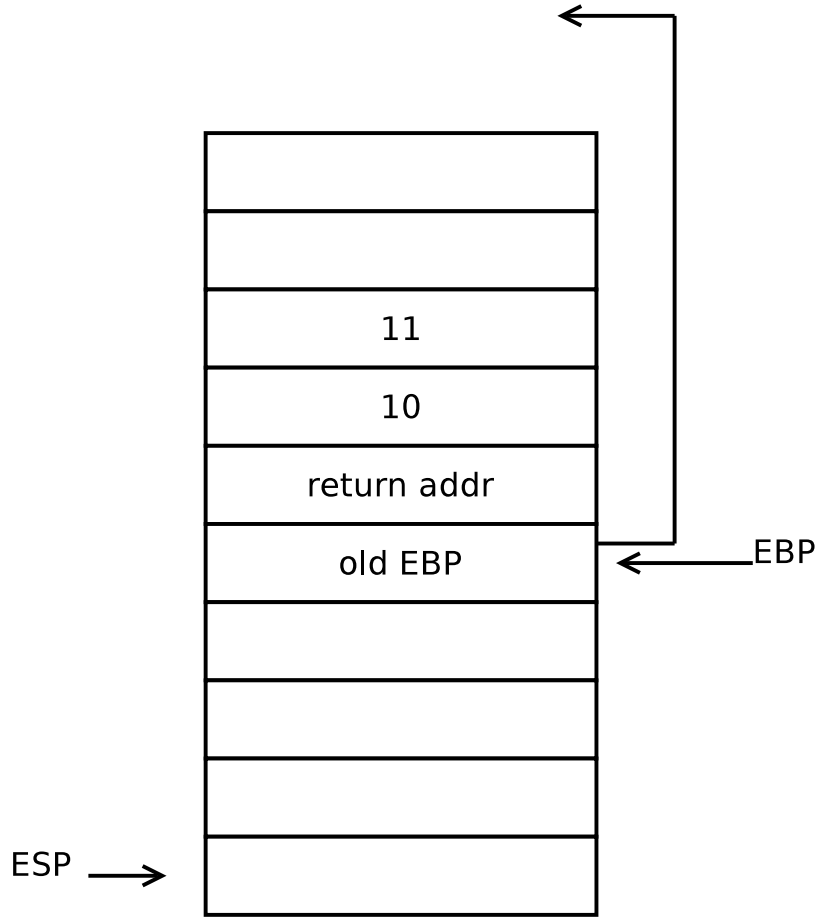
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

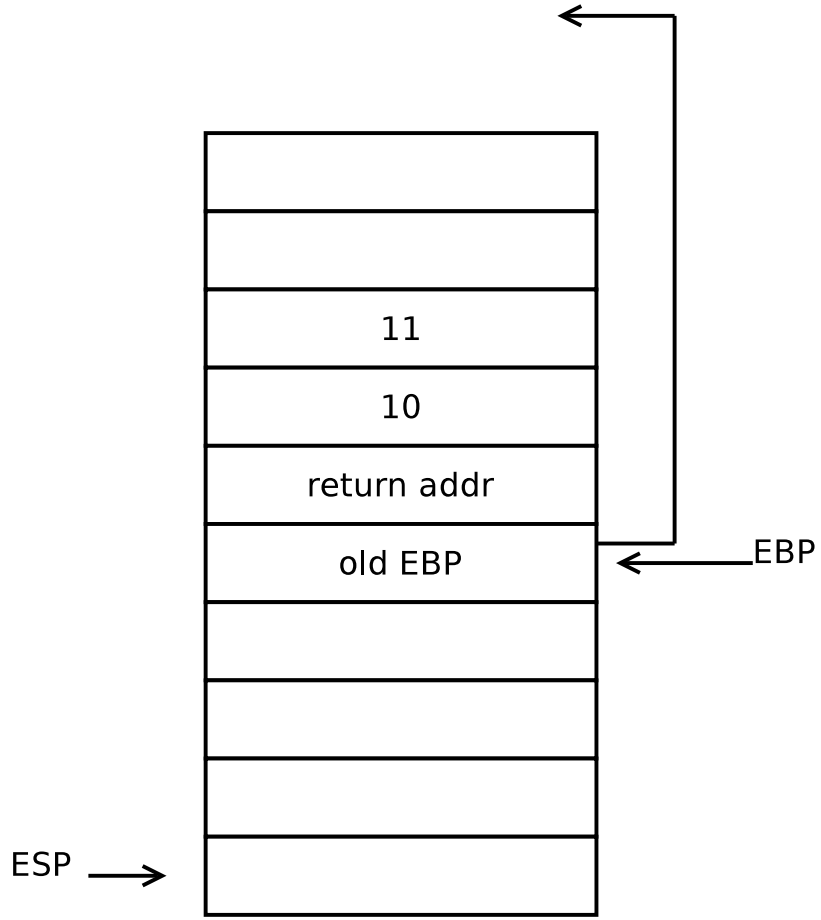
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

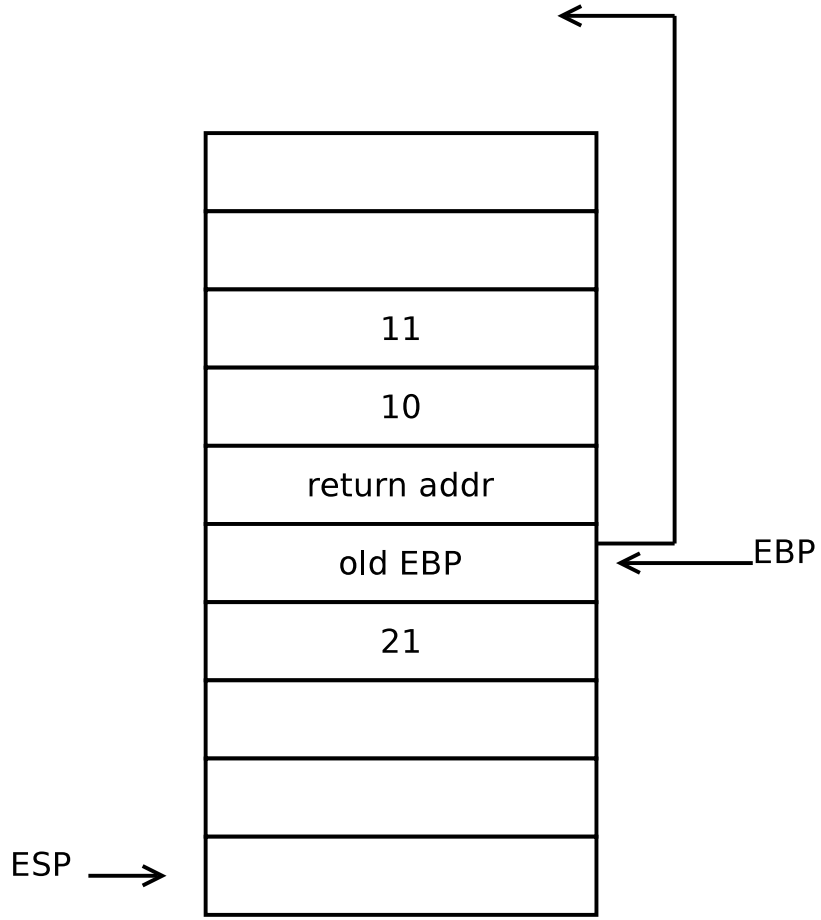
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

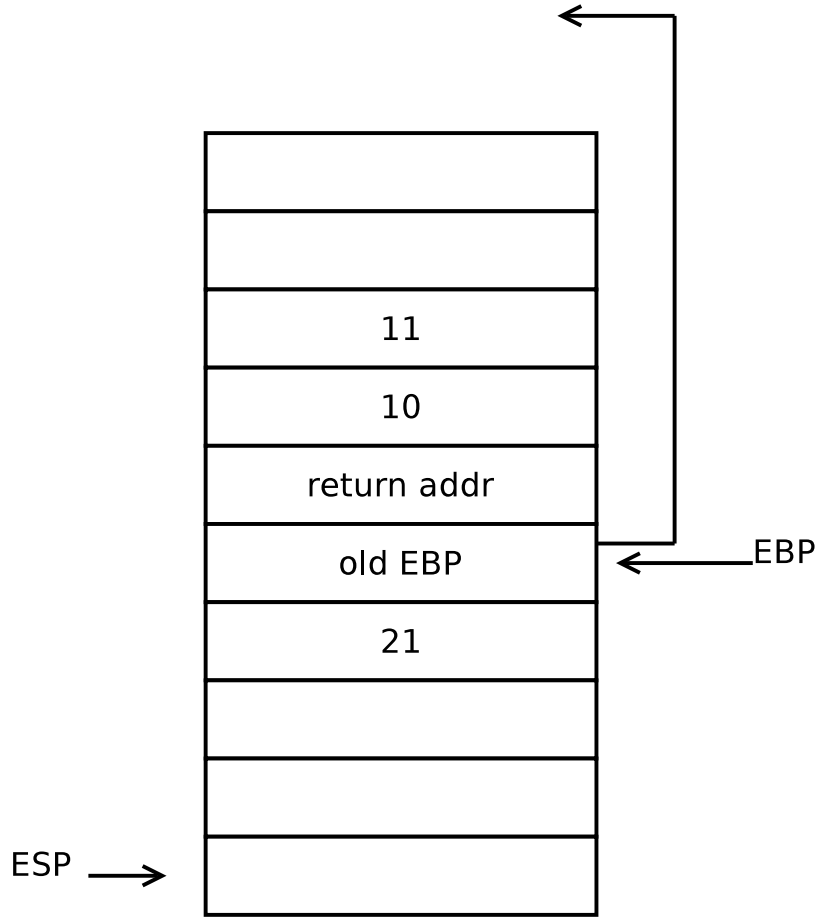
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

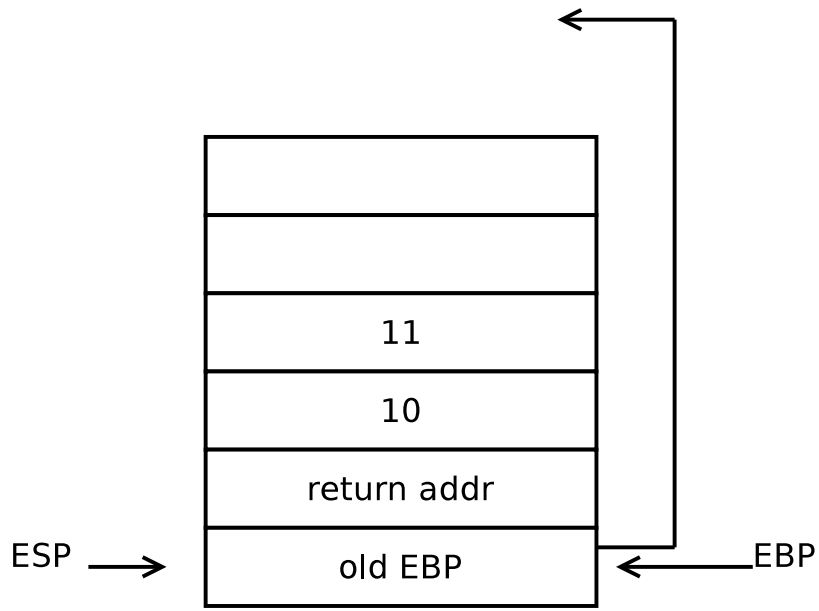
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

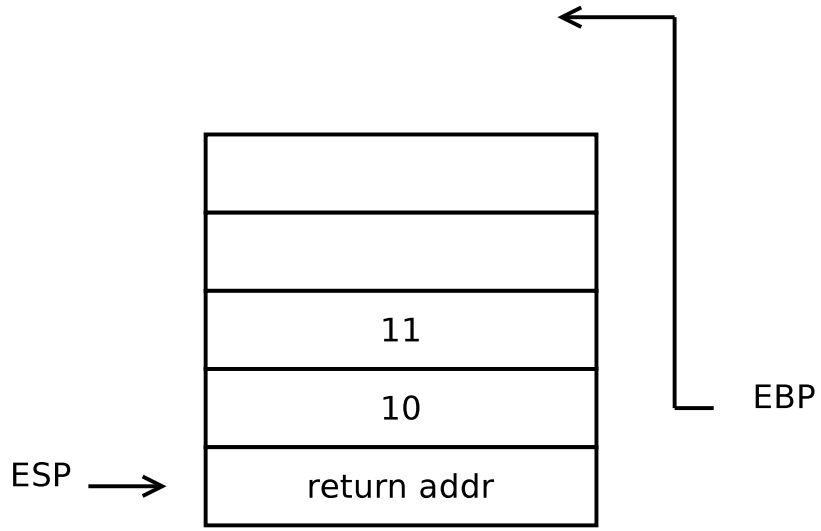
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

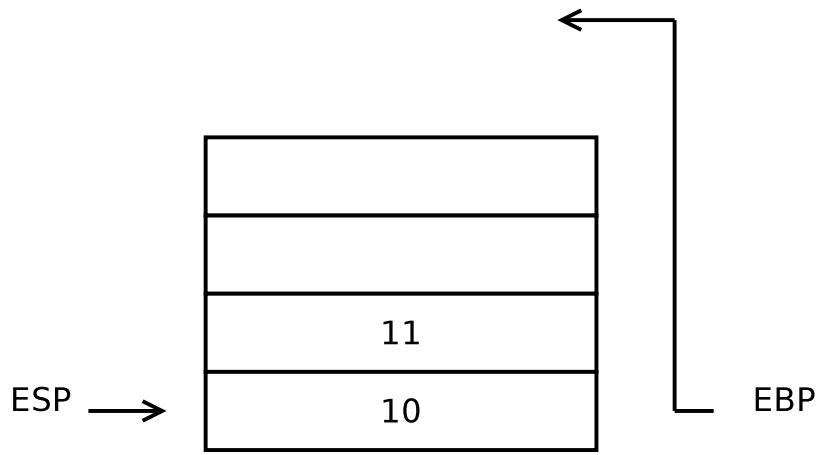
```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



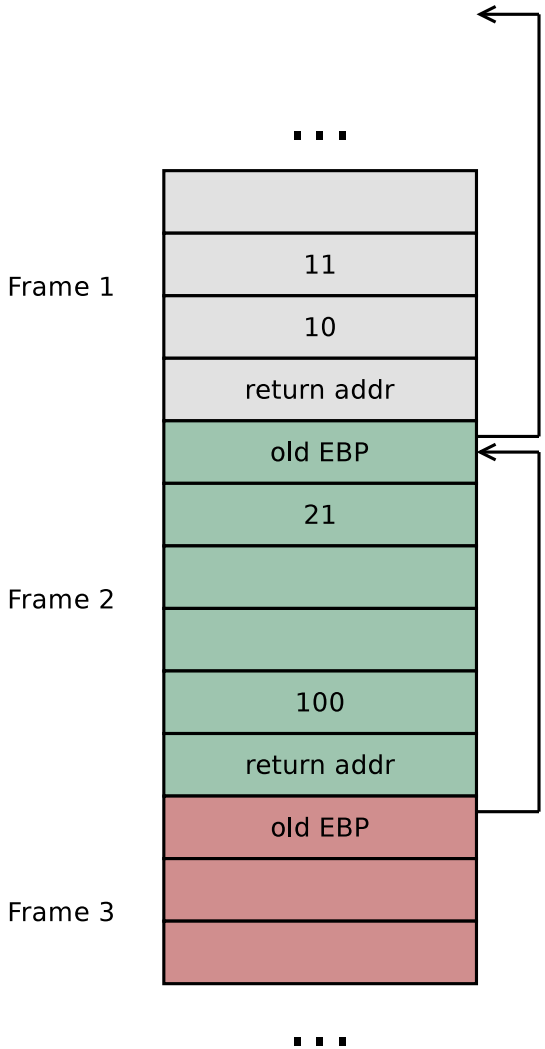
Stack Frames

```c
#include <stdio.h>

int g(int x, int y)
{
        int s;

        s = x+y;

        return s;
}

void f(void)
{
        g(10, 11);
}

int main(void)
{
        f();
}
```

```
g:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0xc(%ebp),%eax
        mov    0x8(%ebp),%edx
        add    %edx,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

f:
        push   %ebp
        mov    %esp,%ebp
        sub    $0x8,%esp
        movl   $0xb,0x4(%esp)
        movl   $0xa,(%esp)
        call   80483d4 <g>
        leave
        ret
```



Stack Frames



Stack Corruption

```
int x = 100;
int v[5] = { 0, 10, 20, 30, 40 };

v[5] = ...;
v[7] = ...;
```