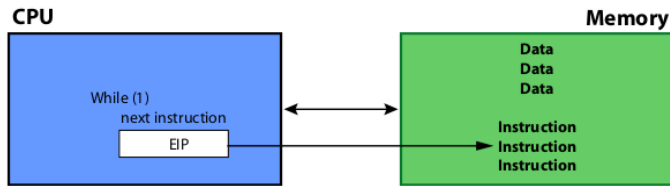


# Memory management

Myrto Arapinis  
School of Informatics  
University of Edinburgh

# From source code to execution

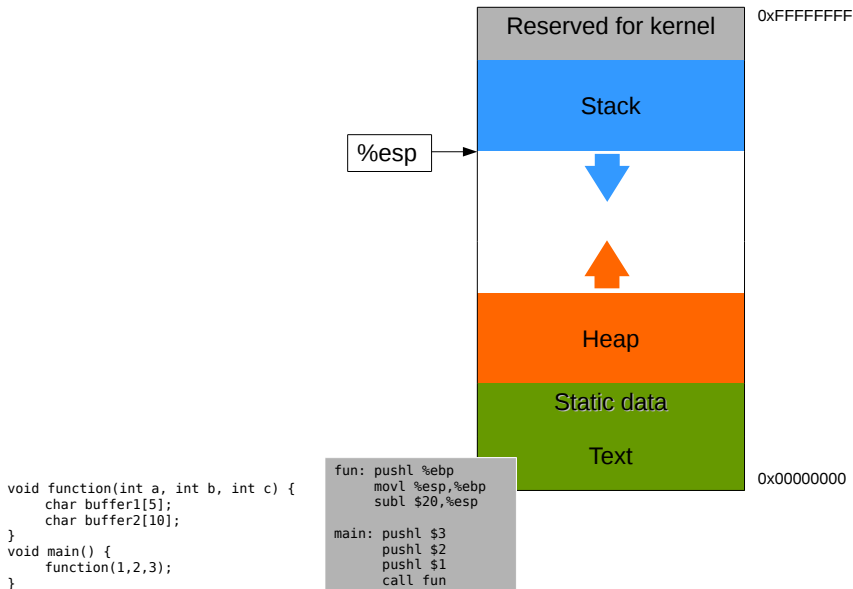


1. The **compiler** converts C code to assembly code
2. The **assembler** converts assembly code to machine code
3. The **linker** deals with dependencies and libraries
4. The **loader** sets up address space in memory and load machine code in memory, and jumps to the first instruction of the program
  - ▶ The address space contains both the code for running the program its input data, and it working memory
5. The **CPU** interprets instructions
  - ▶ `%eip` points to next instruction
  - ▶ `%eip` incremented after each instruction
  - ▶ `%eip` modified by `call`, `ret`, `jmp`, and conditional `jmp`

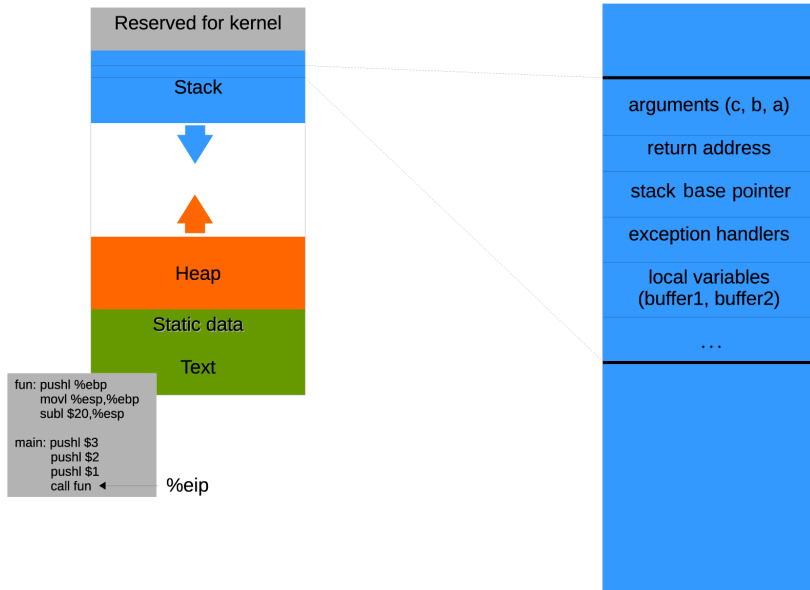
# x86-32 registers

- ▶ Temporary registers: `%eax, %ebx, %ecx, %edx, %edi, %esi`
  - These registers are like variables built in the processor
  - Most of the instructions perform on these registers
- ▶ Extended stack pointer: `%esp`
  - Points at the top of the stack
- ▶ Extended base pointer: `%ebp`
  - Points to the base of the stack frame of the current function call

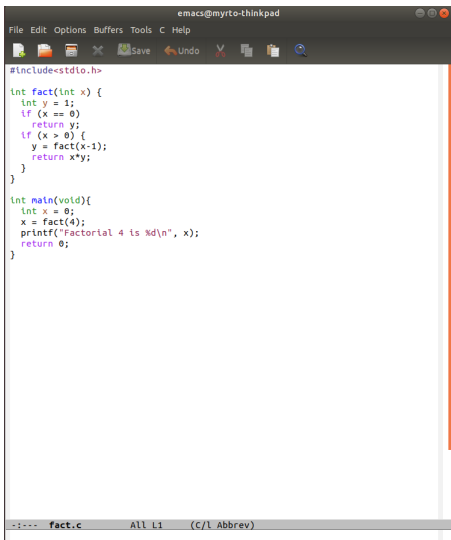
# x86 process memory layout (simplified)



# Stack frame



# x86 assembly - an example



The screenshot shows an Emacs editor window titled 'emacs@myrto-thinkpad'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. The toolbar shows icons for saving, undo, redo, and other editing functions. The main text area contains the following C code:

```
#include<stdio.h>

int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void){
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d\n", x);
    return 0;
}
```

The status bar at the bottom shows 'fact.c', 'All L1', and '(C/L Abbrev)'.



The screenshot shows an Emacs editor window titled 'emacs@myrto-thinkpad'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Asm', and 'Help'. The toolbar shows icons for saving, undo, redo, and other editing functions. The main text area contains the following assembly code:

```
fact:
    pushl    %ebp
    movl     %esp, %ebp
    subl     $40, %esp
    movl     $1, -12(%ebp)
    cmpl     $0, 8(%ebp)
    jne      .L2
    movl     -12(%ebp), %eax
    jmp      .L1

.L2:
    jle      .L1
    movl     8(%ebp), %eax
    subl     $1, %eax
    movl     %eax, (%esp)
    call     fact
    movl     %eax, -12(%ebp)
    movl     8(%ebp), %eax
    tmull    -12(%ebp), %eax
    jmp      .L1

.L1:
    leave
    ret

.LC0:
    .string "Factorial 4 is %d\n"

main:
    pushl    %ebp
    movl     %esp, %ebp
    andl     $-16, %esp
    subl     $32, %esp
    movl     $0, 28(%esp)
    movl     $4, (%esp)
    call     fact
    movl     %eax, 28(%esp)
    movl     28(%esp), %eax
    movl     %eax, 4(%esp)
    movl     $.LC0, (%esp)
    call     printf
    movl     $0, %eax
    leave
    ret

:--- fact.s      All L26      (Assembler)
Wrote /home/narapint/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Lectures/Lec18_OS_Intro/GDB_demo/fact.s
```

The status bar at the bottom shows 'fact.s', 'All L26', and '(Assembler)'. Below the status bar, there is a message: 'Wrote /home/narapint/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Lectures/Lec18\_OS\_Intro/GDB\_demo/fact.s'.

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	← %ebp
SF <sub>main</sub> —	0xbffff4 :	0x00000000	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	← %ebp
SF <sub>main</sub> —	0xbffff4 : 0xbffff0 :	0x00000000 0x00000004	← %esp



# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	← %ebp
SF <sub>main</sub> —	0xbffff4 : 0xbffff0 : 0xbfffec :	0x00000000 0x00000004 0x08048474 @ret_m	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xb7e31a83 @ret0	
	0xbffff8 : 0x00000000 %ebp0	← %ebp
SF <sub>main</sub> —	0xbffff4 : 0x00000000	
	0xbffff0 : 0x00000004	
	0xbfffeec : 0x08048474 @retm	
	0xbfffe8 : 0xbfffe8 %ebp <sub>m</sub>	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	
SF <sub>main</sub> —	0xbffff4 : 0xbffff0 : 0xbfffec : 0xbfffe8 :	0x00000000 0x00000004 0x08048474 @ret <sub>m</sub> 0xbfffe8 %ebp <sub>m</sub>	✓ %ebp ← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	
$SF_{main}$ —	0xbffff4 : 0xbffff0 : 0xbfffeec : 0xbfffe8 :	0x00000000 0x00000004 0x08048474 @ret_m 0xbfffe8 %ebp_m	← %ebp
$SF_{fact(4)}$ —	0xbfffe4 :	0x00000001	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0	
	0xbffffef8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffffef4 :	0x00000000	
	0xbffffef0 :	0x00000004	
	0xbffffefc :	0x08048474 @ret_m	
	0xbffffef8 :	0xbffffef8 %ebp_m	← %ebp
$SF_{fact(4)}$ —	0xbffffefe4 :	0x00000001	
	0xbffffefe0 :	0x00000003	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	
$SF_{main}$ —	0xbffff4 : 0xbffff0 : 0xbfffeec : 0xbfffe8 :	0x00000000 0x00000004 0x08048474 @ret_m 0xbfffe8 %ebp_m	← %ebp
$SF_{fact(4)}$ —	0xbfffe4 : 0xbfffe0 : 0xbfffdc :	0x00000001 0x00000003 0x08048449 @ret_4	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0	
	0xbffffef8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffffef4 :	0x00000000	
	0xbffffef0 :	0x00000004	
	0xbffffefc :	0x08048474 @ret_m	
	0xbffffef8 :	0xbffffef8 %ebp_m	← %ebp
$SF_{fact(4)}$ —	0xbffffefe4 :	0x00000001	
	0xbffffefe0 :	0x00000003	
	0xbffffefd4 :	0x08048449 @ret_4	
	0xbffffefd8 :	0xbffffefe8 %ebp_4	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0	
	0xbffffef8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffffef4 :	0x00000000	
	0xbffffef0 :	0x00000004	
	0xbffffefc :	0x08048474 @ret <sub>m</sub>	
	0xbffffef8 :	0xbffffef8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbffffefe4 :	0x00000001	
	0xbffffefe0 :	0x00000003	
	0xbffffefd4 :	0x08048449 @ret <sub>4</sub>	✓ %ebp
	0xbffffefd8 :	0xbffffefe8 %ebp <sub>4</sub>	← %esp



# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0	
$SF_{main}$ —	0xbffff4 : 0xbffff0 : 0xbfffec : 0xbfffe8 :	0x00000000 0x00000004 0x08048474 @ret_m 0xbfffe8 %ebp_m	
$SF_{fact(4)}$ —	0xbfffe4 : 0xbfffe0 : 0xbfffdc : 0xbfffd8 :	0x00000001 0x00000003 0x08048449 @ret_4 0xbfffe8 %ebp_4	← %ebp
$SF_{fact(3)}$ —	0xbfffd4 :	0x00000001	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0	
	0xbffffeff8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffffeff4 :	0x00000000	
	0xbffffeff0 :	0x00000004	
	0xbffffefec :	0x08048474 @ret <sub>m</sub>	
	0xbffffefe8 :	0xbffffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbffffefe4 :	0x00000001	
	0xbffffefe0 :	0x00000003	
	0xbffffefd4 :	0x08048449 @ret <sub>4</sub>	
	0xbffffefd8 :	0xbffffefe8 %ebp <sub>4</sub>	← %ebp
$SF_{fact(3)}$ —	0xbffffefd4 :	0x00000001	
	0xbffffefd0 :	0x00000002	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0	
	0xbffffeff8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffffeff4 :	0x00000000	
	0xbffffeff0 :	0x00000004	
	0xbffffefec :	0x08048474 @ret <sub>m</sub>	
	0xbffffefe8 :	0xbffffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbffffefe4 :	0x00000001	
	0xbffffefe0 :	0x00000003	
	0xbffffefdc :	0x08048449 @ret <sub>4</sub>	
	0xbffffefd8 :	0xbffffefe8 %ebp <sub>4</sub>	← %ebp
$SF_{fact(3)}$ —	0xbffffefd4 :	0x00000001	
	0xbffffefd0 :	0x00000002	
	0xbffffefcc :	0x08048449 @ret <sub>3</sub>	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0	
	0xbffffc8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffffc4 :	0x00000000	
	0xbffffc0 :	0x00000004	
	0xbffffec :	0x08048474 @ret <sub>m</sub>	
	0xbffffc8 :	0xbffffc8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbffffe4 :	0x00000001	
	0xbffffe0 :	0x00000003	
	0xbffffdc :	0x08048449 @ret <sub>4</sub>	
	0xbffffd8 :	0xbffffe8 %ebp <sub>4</sub>	← %ebp
$SF_{fact(3)}$ —	0xbffffd4 :	0x00000001	
	0xbffffd0 :	0x00000002	
	0xbffffcc :	0x08048449 @ret <sub>3</sub>	
	0xbffffc8 :	0xbffffd8 %ebp <sub>3</sub>	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0
	0xbffffc8 :	0x00000000 %ebp0
<i>SF<sub>main</sub></i> —	0xbffffc4 :	0x00000000
	0xbffffc0 :	0x00000004
	0xbffffec :	0x08048474 @ret <sub>m</sub>
	0xbffffc8 :	0xbffffc8 %ebp <sub>m</sub>
<i>SF<sub>fact(4)</sub></i> —	0xbffffe4 :	0x00000001
	0xbffffe0 :	0x00000003
	0xbffffdc :	0x08048449 @ret <sub>4</sub>
	0xbffffd8 :	0xbffffe8 %ebp <sub>4</sub>
<i>SF<sub>fact(3)</sub></i> —	0xbffffd4 :	0x00000001
	0xbffffd0 :	0x00000002
	0xbffffcc :	0x08048449 @ret <sub>3</sub>
	0xbffffc8 :	0xbffffd8 %ebp <sub>3</sub>

✓ %ebp  
← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0
$SF_{main}$ —	0xbffff4 : 0xbffff0 : 0xbfffec : 0xbfffe8 :	0x00000000 0x00000004 0x08048474 @ret_m 0xbffff8 %ebp_m
$SF_{fact(4)}$ —	0xbfffe4 : 0xbfffe0 : 0xbfffdc : 0xbfffd8 :	0x00000001 0x00000003 0x08048449 @ret_4 0xbfffe8 %ebp_4
$SF_{fact(3)}$ —	0xbfffd4 : 0xbfffd0 : 0xbfffcc : 0xbfffc8 :	0x00000001 0x00000002 0x08048449 @ret_3 0xbfffd8 %ebp_3
$SF_{fact(2)}$ —	0xbfffc4 :    	0x00000001    

← %ebp

← %esp

%eax

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbfffeffc :	0xb7e31a83 @ret0	
	0xbfffeff8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000002	
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>	← %ebp
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffecfc :	0x00000001	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0
	0xbffffc8 :	0x00000000 %ebp0
$SF_{main}$ —	0xbffffc4 :	0x00000000
	0xbffffc0 :	0x00000004
	0xbffffec :	0x08048474 @ret <sub>m</sub>
	0xbffffc8 :	0xbffffc8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffffc4 :	0x00000001
	0xbffffc0 :	0x00000003
	0xbffffdc :	0x08048449 @ret <sub>4</sub>
	0xbffffd8 :	0xbffffc8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffffd4 :	0x00000001
	0xbffffd0 :	0x00000002
	0xbffffcc :	0x08048449 @ret <sub>3</sub>
	0xbffffc8 :	0xbffffd8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffffc4 :	0x00000001
	0xbffffc0 :	0x00000001
	0xbffffbc :	0x08048449 @ret <sub>2</sub>

← %ebp

← %esp



# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffcfc :	0xb7e31a83 @ret0
	0xbffffc78 :	0x00000000 %ebp0
$SF_{main}$ —	0xbffffc74 :	0x00000000
	0xbffffc70 :	0x00000004
	0xbffffc6c :	0x08048474 @ret <sub>m</sub>
	0xbffffc68 :	0xbffffc78 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffffc64 :	0x00000001
	0xbffffc60 :	0x00000003
	0xbffffc5c :	0x08048449 @ret <sub>4</sub>
	0xbffffc58 :	0xbffffc68 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffffc54 :	0x00000001
	0xbffffc50 :	0x00000002
	0xbffffc4c :	0x08048449 @ret <sub>3</sub>
	0xbffffc48 :	0xbffffc58 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffffc44 :	0x00000001
	0xbffffc40 :	0x00000001
	0xbffffc3c :	0x08048449 @ret <sub>2</sub>
	0xbffffc38 :	0xbffffc48 %ebp <sub>2</sub>

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffefffc :	0xb7e31a83 @ret0
	0xbffefff8 :	0x00000000 %ebp0
$SF_{main}$ —	0xbffefff4 :	0x00000000
	0xbffefff0 :	0x00000004
	0xbffeffec :	0x08048474 @ret <sub>m</sub>
	0xbffeffe8 :	0xbffefff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffefe4 :	0x00000001
	0xbffefe0 :	0x00000003
	0xbffefdc :	0x08048449 @ret <sub>4</sub>
	0xbffefd8 :	0xbffefe8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffefd4 :	0x00000001
	0xbffefd0 :	0x00000002
	0xbffefcc :	0x08048449 @ret <sub>3</sub>
	0xbffefc8 :	0xbffefd8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffefc4 :	0x00000001
	0xbffefc0 :	0x00000001
	0xbffefbc :	0x08048449 @ret <sub>2</sub>
	0xbffefb8 :	0xbffefc8 %ebp <sub>2</sub>

↙ %ebp  
← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffefffc :	0xb7e31a83 @ret0
	0xbffefff8 :	0x00000000 %ebp0
$SF_{main}$ —	0xbffefff4 :	0x00000000
	0xbffefff0 :	0x00000004
	0xbffeffec :	0x08048474 @ret <sub>m</sub>
	0xbffeffe8 :	0xbffefff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffefe4 :	0x00000001
	0xbffefe0 :	0x00000003
	0xbffefdc :	0x08048449 @ret <sub>4</sub>
	0xbffefd8 :	0xbffefe8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffefd4 :	0x00000001
	0xbffefd0 :	0x00000002
	0xbffefcc :	0x08048449 @ret <sub>3</sub>
	0xbffefc8 :	0xbffefd8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffefc4 :	0x00000001
	0xbffefc0 :	0x00000001
	0xbffefbc :	0x08048449 @ret <sub>2</sub>
	0xbffefb8 :	0xbffefc8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbffefb4 :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffefffc :	0xb7e31a83 @ret0	
	0xbffefff8 :	0x00000000 %ebp0	
$SF_{main}$ —	0xbffefff4 :	0x00000000	
	0xbffefff0 :	0x00000004	
	0xbffeffec :	0x08048474 @ret <sub>m</sub>	
	0xbffeffe8 :	0xbffefff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbffefe4 :	0x00000001	
	0xbffefe0 :	0x00000003	
	0xbffefdc :	0x08048449 @ret <sub>4</sub>	
	0xbffefd8 :	0xbffefe8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbffefd4 :	0x00000001	
	0xbffefd0 :	0x00000002	
	0xbffefcc :	0x08048449 @ret <sub>3</sub>	
	0xbffefc8 :	0xbffefd8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbffefc4 :	0x00000001	
	0xbffefc0 :	0x00000001	
	0xbffefbc :	0x08048449 @ret <sub>2</sub>	
	0xbffefb8 :	0xbffefc8 %ebp <sub>2</sub>	← %ebp
$SF_{fact(1)}$ —	0xbffefb4 :	0x00000001	
	0xbffefb0 :	0x00000000	← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffffc : 0xbffff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0
$SF_{main}$ —	0xbffff4 : 0xbffff0 : 0xbfffeec : 0xbfffe8 :	0x00000000 0x00000004 0x08048474 @ret_m 0xbfffeff8 %ebp_m
$SF_{fact(4)}$ —	0xbfffe4 : 0xbfffe0 : 0xbfffedc : 0xbffefd8 :	0x00000001 0x00000003 0x08048449 @ret_4 0xbfffe8 %ebp_4
$SF_{fact(3)}$ —	0xbffefd4 : 0xbffefd0 : 0xbffefcc : 0xbffefc8 :	0x00000001 0x00000002 0x08048449 @ret_3 0xbffefd8 %ebp_3
$SF_{fact(2)}$ —	0xbffefc4 : 0xbffefc0 : 0xbffefbc : 0xbffefb8 :	0x00000001 0x00000001 0x08048449 @ret_2 0xbffefc8 %ebp_2
$SF_{fact(1)}$ —	0xbffefb4 : 0xbffefb0 : 0xbffefac :	0x00000001 0x00000000 0x08048449 @ret_1

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffec8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>
	0xbfffeb8 :	0xbfffec8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffeb4 :	0x00000001
	0xbfffeb0 :	0x00000000
	0xbffefac :	0x08048449 @ret <sub>1</sub>
	0xbffefa8 :	0xbfffeb8 %ebp <sub>1</sub>

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbffeff4 :	0x00000000
	0xbffeff0 :	0x00000004
	0xbffefec :	0x08048474 @ret <sub>m</sub>
	0xbffefe8 :	0xbffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffefe4 :	0x00000001
	0xbffefe0 :	0x00000003
	0xbffefdc :	0x08048449 @ret <sub>4</sub>
	0xbffefd8 :	0xbffefe8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffefd4 :	0x00000001
	0xbffefd0 :	0x00000002
	0xbffefcc :	0x08048449 @ret <sub>3</sub>
	0xbffefc8 :	0xbffefd8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffefc4 :	0x00000001
	0xbffefc0 :	0x00000001
	0xbffefbc :	0x08048449 @ret <sub>2</sub>
	0xbffefb8 :	0xbffefc8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbffefb4 :	0x00000001
	0xbffefb0 :	0x00000000
	0xbffefac :	0x08048449 @ret <sub>1</sub>
	0xbffefa8 :	0xbffefb8 %ebp <sub>1</sub>

↙ %ebp  
← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000000
	0xbfffecac :	0x08048449 @ret <sub>1</sub>
	0xbfffeffa :	0xbfffeff8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbfffeff4 :	0x00000001

← %ebp

← %esp



# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000000
	0xbfffecac :	0x08048449 @ret <sub>1</sub>
	0xbfffeffa :	0xbfffeff8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbfffeffa :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000000
	0xbfffecac :	0x08048449 @ret <sub>1</sub>
	0xbfffeffa :	0xbfffeff8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbfffeffa :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000000
	0xbfffecac :	0x08048449 @ret <sub>1</sub>
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbfffeff4 :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbffeffc : 0xbffeff8 :	... 0xb7e31a83 @ret0 0x00000000 %ebp0
$SF_{main}$ —	0xbffeff4 : 0xbffeff0 : 0xbffefec : 0xbffefe8 :	0x00000000 0x00000004 0x08048474 @ret_m 0xbffeff8 %ebp_m
$SF_{fact(4)}$ —	0xbffefe4 : 0xbffefe0 : 0xbffefdc : 0xbffefd8 :	0x00000001 0x00000003 0x08048449 @ret_4 0xbffefe8 %ebp_4
$SF_{fact(3)}$ —	0xbffefd4 : 0xbffefd0 : 0xbffefcc : 0xbffefc8 :	0x00000001 0x00000002 0x08048449 @ret_3 0xbffefd8 %ebp_3
$SF_{fact(2)}$ —	0xbffefc4 : 0xbffefc0 : 0xbffefbc : 0xbffefb8 :	0x00000001 0x00000001 0x08048449 @ret_2 0xbffefc8 %ebp_2
$SF_{fact(1)}$ —	0xbffefb4 : 0xbffefb0 : 0xbffefac : 0xbffefa8 :	0x00000001 0x00000000 0x08048449 @ret_1 0xbffefb8 %ebp_1
$SF_{fact(0)}$ —	0xbffefa4 : 	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffec8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>
	0xbfffeb8 :	0xbfffec8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffebfc :	0x00000001
	0xbfffeb8 :	0x00000000
	0xbffefac :	0x08048449 @ret <sub>1</sub>
	0xbffef8 :	0xbfffeb8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbffefa4 :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbffeff4 :	0x00000000
	0xbffeff0 :	0x00000004
	0xbffefec :	0x08048474 @ret <sub>m</sub>
	0xbffefe8 :	0xbffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffefe4 :	0x00000001
	0xbffefe0 :	0x00000003
	0xbffefdc :	0x08048449 @ret <sub>4</sub>
	0xbffefd8 :	0xbffefe8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffefd4 :	0x00000001
	0xbffefd0 :	0x00000002
	0xbffefcc :	0x08048449 @ret <sub>3</sub>
	0xbffefc8 :	0xbffefd8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffefc4 :	0x00000001
	0xbffefc0 :	0x00000001
	0xbffefbc :	0x08048449 @ret <sub>2</sub>
	0xbffefb8 :	0xbffefc8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbffefb4 :	0x00000001
	0xbffefb0 :	0x00000000
	0xbffefac :	0x08048449 @ret <sub>1</sub>
	0xbffefa8 :	0xbffefb8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbffefa4 :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbffeff4 :	0x00000000
	0xbffeff0 :	0x00000004
	0xbffefec :	0x08048474 @ret <sub>m</sub>
	0xbffefe8 :	0xbffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbffefe4 :	0x00000001
	0xbffefe0 :	0x00000003
	0xbffefdc :	0x08048449 @ret <sub>4</sub>
	0xbffefd8 :	0xbffefe8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbffefd4 :	0x00000001
	0xbffefd0 :	0x00000002
	0xbffefcc :	0x08048449 @ret <sub>3</sub>
	0xbffefc8 :	0xbffefd8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbffefc4 :	0x00000001
	0xbffefc0 :	0x00000001
	0xbffefbc :	0x08048449 @ret <sub>2</sub>
	0xbffefb8 :	0xbffefc8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbffefb4 :	0x00000001
	0xbffefb0 :	0x00000000
	0xbffefac :	0x08048449 @ret <sub>1</sub>
	0xbffefa8 :	0xbffefb8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbffefa4 :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000001

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000002
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000000
	0xbfffecac :	0x08048449 @ret <sub>1</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbfffeff4 :	0x00000001

← %ebp

← %esp



# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000002

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffedfc :	0x08048449 @ret <sub>4</sub>
	0xbfffedf8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffedf4 :	0x00000001
	0xbfffedf0 :	0x00000002
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>
	0xbfffedf8 :	0xbfffedf8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>
	0xbfffebfb :	0xbfffecf8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffebfb :	0x00000001
	0xbfffebfb :	0x00000000
	0xbfffebfa :	0x08048449 @ret <sub>1</sub>
	0xbfffebfa :	0xbfffebfb %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbfffebfa :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000002

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>
$SF_{main}$ —	0xbfffeff4 :	0x00000000
	0xbfffeff0 :	0x00000004
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001
	0xbfffeff0 :	0x00000003
	0xbfffedfc :	0x08048449 @ret <sub>4</sub>
	0xbfffedf8 :	0xbfffeff8 %ebp <sub>4</sub>
$SF_{fact(3)}$ —	0xbfffedf4 :	0x00000001
	0xbfffedf0 :	0x00000002
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>
	0xbfffedf8 :	0xbfffedf8 %ebp <sub>3</sub>
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001
	0xbfffecf0 :	0x00000001
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>
	0xbfffebfb :	0xbfffecf8 %ebp <sub>2</sub>
$SF_{fact(1)}$ —	0xbfffebfb :	0x00000001
	0xbfffebfb :	0x00000000
	0xbffefbac :	0x08048449 @ret <sub>1</sub>
	0xbffefba8 :	0xbfffebfb %ebp <sub>1</sub>
$SF_{fact(0)}$ —	0xbffefba4 :	0x00000001

← %ebp

← %esp

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000002

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000003	
	0xbfffedfc :	0x08048449 @ret <sub>4</sub>	
	0xbfffedf8 :	0xbfffeff8 %ebp <sub>4</sub>	← %ebp
$SF_{fact(3)}$ —	0xbfffedf4 :	0x00000001	
	0xbfffedf0 :	0x00000002	
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>	
	0xbfffecf8 :	0xbfffedf8 %ebp <sub>3</sub>	← %esp
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>	
	0xbfffebfb :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffebfb :	0x00000001	
	0xbfffebfb :	0x00000000	
	0xbfffebfa :	0x08048449 @ret <sub>1</sub>	
	0xbfffebfa :	0xbfffebfb %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffebfa :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000002

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000003	
	0xbfffedfc :	0x08048449 @ret <sub>4</sub>	
	0xbfffedf8 :	0xbfffeff8 %ebp <sub>4</sub>	← %ebp
$SF_{fact(3)}$ —	0xbfffedf4 :	0x00000002	
	0xbfffedf0 :	0x00000002	
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>	
	0xbfffecf8 :	0xbfffedf8 %ebp <sub>3</sub>	← %esp
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>	
	0xbfffebfb8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffebfb4 :	0x00000001	
	0xbfffebfb0 :	0x00000000	
	0xbfffebfa8 :	0x08048449 @ret <sub>1</sub>	
	0xbfffebfa8 :	0xbfffebfb8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffebfa4 :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000006

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000003	
	0xbfffedfc :	0x08048449 @ret <sub>4</sub>	
	0xbfffedf8 :	0xbfffeff8 %ebp <sub>4</sub>	← %ebp
$SF_{fact(3)}$ —	0xbfffedf4 :	0x00000002	
	0xbfffedf0 :	0x00000002	
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>	
	0xbfffecf8 :	0xbfffedf8 %ebp <sub>3</sub>	← %esp
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbffefbfc :	0x08048449 @ret <sub>2</sub>	
	0xbffefb8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbffefb4 :	0x00000001	
	0xbffefb0 :	0x00000000	
	0xbffefac :	0x08048449 @ret <sub>1</sub>	
	0xbffefaf8 :	0xbffefb8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbffefa4 :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000006

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	← %ebp
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000003	
	0xbfffeffc :	0x08048449 @ret <sub>4</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000002	
	0xbfffeff0 :	0x00000002	
	0xbfffecfc :	0x08048449 @ret <sub>3</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>	← %esp
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000001	
	0xbfffeffc :	0x08048449 @ret <sub>2</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000000	
	0xbfffecfc :	0x08048449 @ret <sub>1</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffeff4 :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000006

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	← %ebp
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffecf8 :	0xbfffeff8 %ebp <sub>4</sub>	← %esp
$SF_{fact(3)}$ —	0xbfffecf4 :	0x00000002	
	0xbfffecf0 :	0x00000002	
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>	
	0xbfffecf8 :	0xbfffecf8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffebfc :	0x08048449 @ret <sub>2</sub>	
	0xbfffecf8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffebfb :	0x00000001	
	0xbfffebfb :	0x00000000	
	0xbfffebfa :	0x08048449 @ret <sub>1</sub>	
	0xbfffebfa :	0xbfffebfb %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffebfa :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000006

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	← %ebp
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000006	← %esp
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000002	
	0xbfffeff0 :	0x00000002	
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>	
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000000	
	0xbfffecac :	0x08048449 @ret <sub>1</sub>	
	0xbfffeffa :	0xbfffeff8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffeffa :	0x00000001	



# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000018

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	← %ebp
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000006	
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>	← %esp
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000002	
	0xbfffeff0 :	0x00000002	
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>	
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000000	
	0xbfffecac :	0x08048449 @ret <sub>1</sub>	
	0xbfffeffa :	0xbfffeff8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffeffa :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000018

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	← %ebp
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000006	← %esp
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffecf8 :	0xbfffeff8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbfffecf4 :	0x00000002	
	0xbfffecf0 :	0x00000002	
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>	
	0xbfffecf8 :	0xbfffecf8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbfffecf4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffecbc :	0x08048449 @ret <sub>2</sub>	
	0xbfffecf8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffecb4 :	0x00000001	
	0xbfffecb0 :	0x00000000	
	0xbfffecac :	0x08048449 @ret <sub>1</sub>	
	0xbfffecf8 :	0xbfffecf8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffecf4 :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000018

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	← %ebp
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000000	← %esp
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000006	
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000002	
	0xbfffeff0 :	0x00000002	
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffeffb :	0x08048449 @ret <sub>2</sub>	
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffeff0 :	0x00000000	
	0xbfffecac :	0x08048449 @ret <sub>1</sub>	
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffeff4 :	0x00000001	

# x86 runtime memory - an example

```
int fact(int x) {
    int y = 1;
    if (x == 0)
        return y;
    if (x > 0) {
        y = fact(x-1);
        return x*y;
    }
}

int main(void) {
    int x = 0;
    x = fact(4);
    printf("Factorial 4 is %d", x);
    return 0;
}
```

%eax

0x00000018

	0xbfffeffc :	0xb7e31a83 @ret <sub>0</sub>	← %ebp
	0xbfffeff8 :	0x00000000 %ebp <sub>0</sub>	
$SF_{main}$ —	0xbfffeff4 :	0x00000018	← %esp
	0xbfffeff0 :	0x00000004	
	0xbfffecfc :	0x08048474 @ret <sub>m</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>m</sub>	
$SF_{fact(4)}$ —	0xbfffeff4 :	0x00000006	
	0xbfffeff0 :	0x00000003	
	0xbfffecdc :	0x08048449 @ret <sub>4</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>4</sub>	
$SF_{fact(3)}$ —	0xbfffeff4 :	0x00000002	
	0xbfffeff0 :	0x00000002	
	0xbfffeccc :	0x08048449 @ret <sub>3</sub>	
	0xbfffeff8 :	0xbfffeff8 %ebp <sub>3</sub>	
$SF_{fact(2)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffecf0 :	0x00000001	
	0xbfffecbc :	0x08048449 @ret <sub>2</sub>	
	0xbfffeff8 :	0xbfffecf8 %ebp <sub>2</sub>	
$SF_{fact(1)}$ —	0xbfffeff4 :	0x00000001	
	0xbfffecb0 :	0x00000000	
	0xbfffecac :	0x08048449 @ret <sub>1</sub>	
	0xbfffeff8 :	0xbfffecb8 %ebp <sub>1</sub>	
$SF_{fact(0)}$ —	0xbfffeff4 :	0x00000001	

# Stack and functions: Summary

## Calling function

1. Push arguments onto the stack (in reverse)
2. Push the return address, i.e., the address of the instruction to run after control returns
3. Jump to the function's address

## Called function

4. Push the old frame pointer onto the stack (`%ebp`)
5. Set frame pointer (`%ebp`) to where the end of the stack is right now (`%esp`)
6. Push local variables onto the stack

## Returning function

7. Reset the previous stack frame: `%esp = %ebp, %ebp = (%ebp)`
8. Jump back to return address: `%eip = 4(%ebp)`