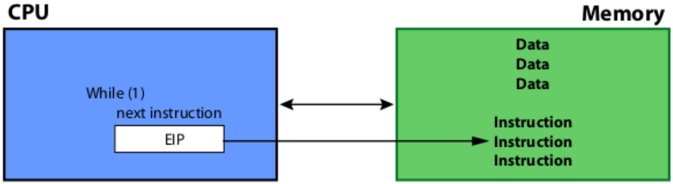
Memory management

Myrto Arapinis
School of Informatics
University of Edinburgh

From source code to execution

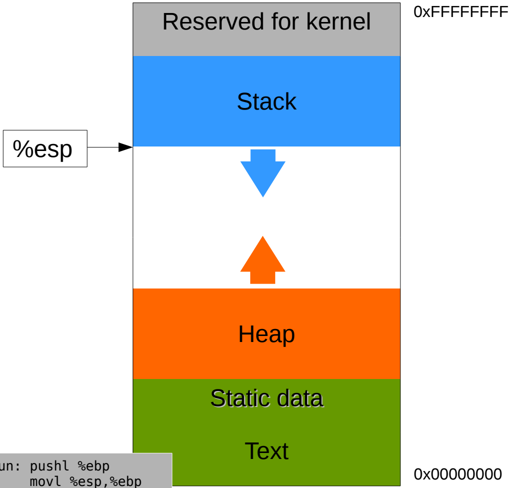


1. The compiler converts C code to assembly code
2. The assembler converts assembly code to machine code
3. The linker deals with dependencies and libraries
4. The loader sets up address space in memory and load machine code in memory, and jumps to the first instruction of the program
   - The address 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

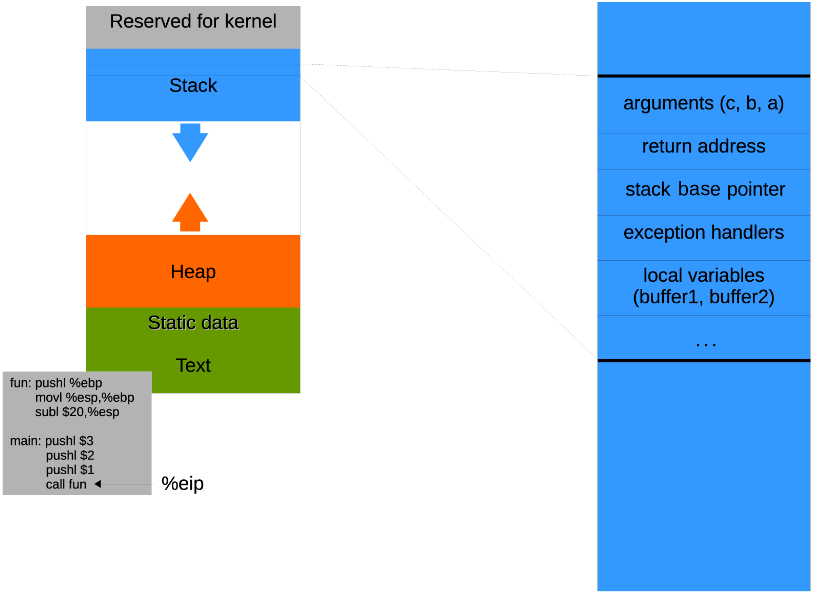
x86 process memory layout (simplified)



```
void function(int a, int b, int c) {
     char buffer1[5];
     char buffer2[10];
}
void main() {
     function(1,2,3);
}
```

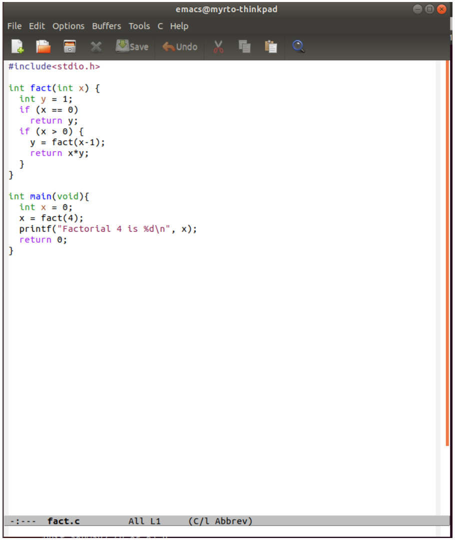
```
fun: pushl %ebp
     movl %esp,%ebp
     subl $20,%esp

main: pushl $3
      pushl $2
      pushl $1
      call fun
```

Stack frame



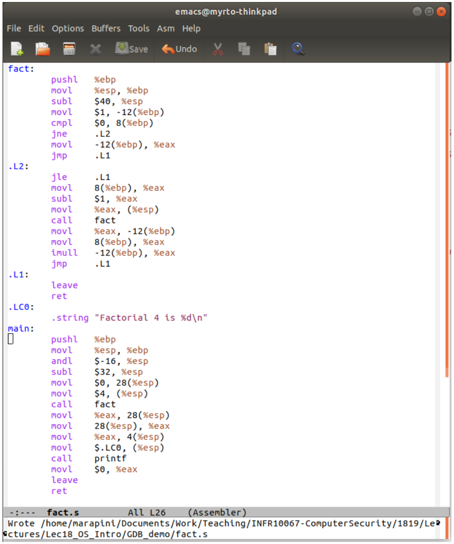
```
fun: pushl %ebp
     movl %esp,%ebp
     subl $20,%esp

main: pushl $3
      pushl $2
      pushl $1
      call fun
```

x86 assembly - an example



```c
#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;
}
```

```
-:---  fact.c        All L1     (C/l Abbrev)
```

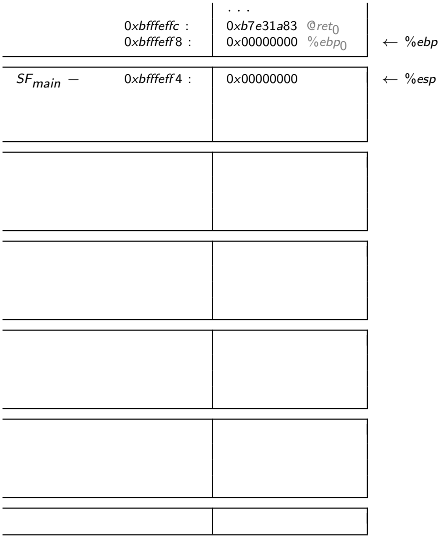


```asm
fact:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $40, %esp
        movl    $1, -12(%ebp)
        cmpl    $0, 8(%ebp)
        jne     .L2
        movl    -12(%ebp), %eax
        jmp     .L1
.L2:
        jle     .L1
        movl    8(%ebp), %eax
        subl    $1, %eax
        movl    %eax, (%esp)
        call    fact
        movl    %eax, -12(%ebp)
        movl    8(%ebp), %eax
        imull   -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/marapini/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Le
ctures/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;
}
```

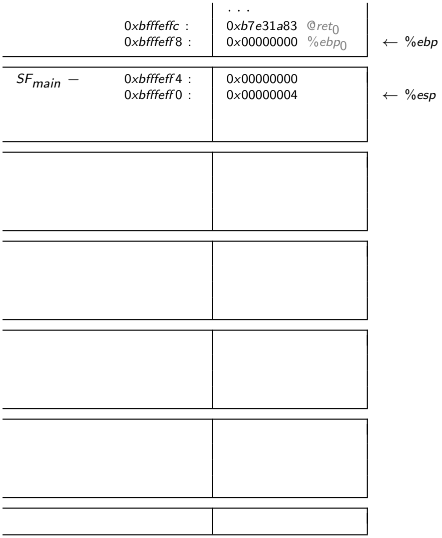


...
0xbfffeffc : 0xb7e31a83 @ret0
0xbfffeff8 : 0x00000000 %ebp0 ← %ebp
SFmain — 0xbfffeff4 : 0x00000000 ← %esp

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;
}
```

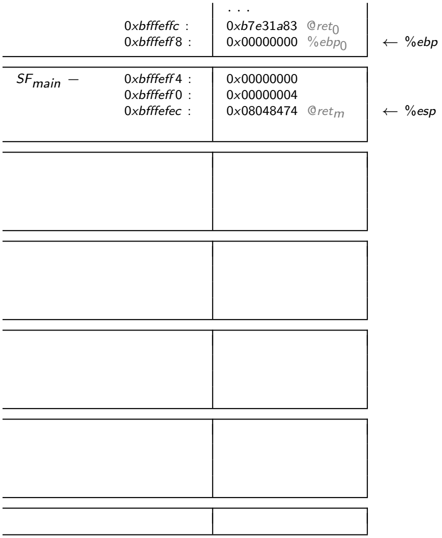
%eax



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;
}
```

$\%eax$

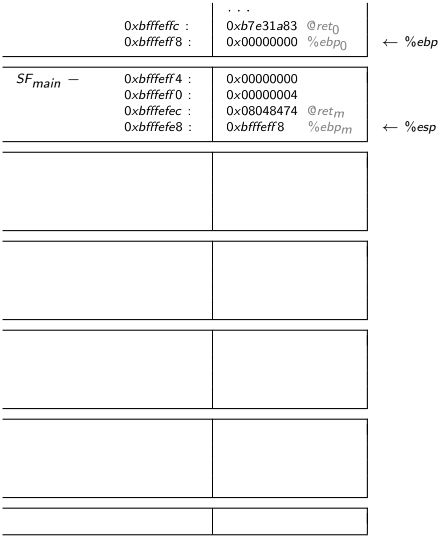


. . .
0xbfffeffc : 0xb7e31a83 @ret_0
0xbfffeff8 : 0x00000000 %ebp_0 ← %ebp
SF_{main} —
0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @ret_m ← %esp

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;
}
```

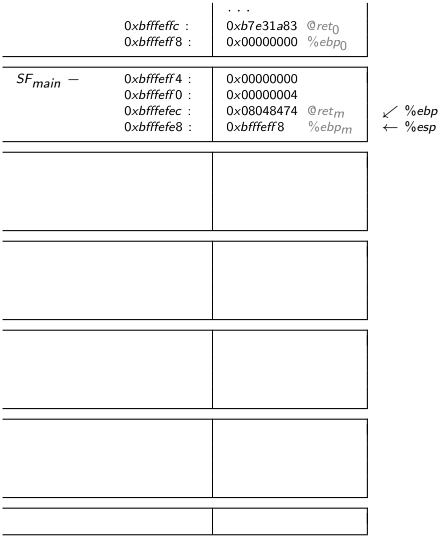
%eax



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	← %ebp
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_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;
}
```

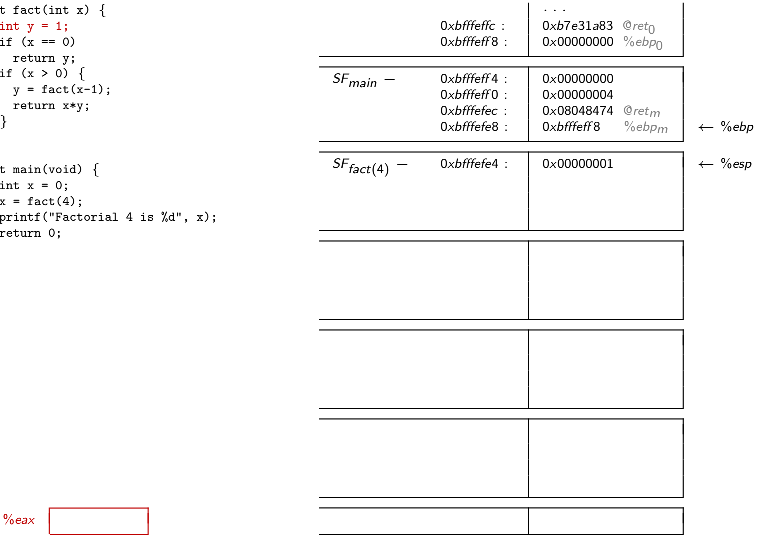


. . .
0xbfffeffc : 0xb7e31a83 @ret0
0xbfffeff8 : 0x00000000 %ebp0
SFmain —
0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @retm
0xbfffefe8 : 0xbfffeff8 %ebpm
%ebp
%esp

%eax

x86 runtime memory - an example

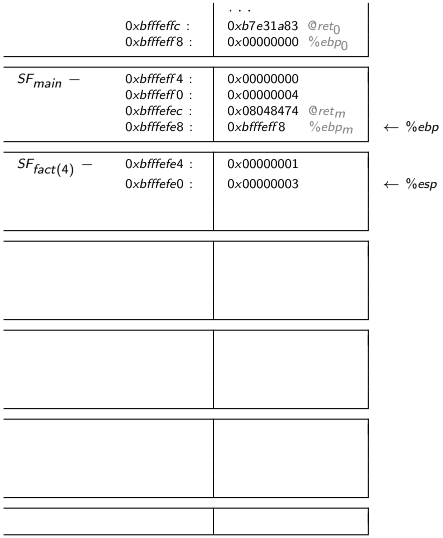
```
int fact(int x) {
  int y = 1;
  if (x == 0)
    return y;
  if (x > 0) {
    y = fact(x-1);
    return x*y;
  }
}

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



. . .
0xbfffeffc : 0xb7e31a83 @ret0
0xbfffeff8 : 0x00000000 %ebp0
SFmain — 0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @retm
0xbfffefe8 : 0xbfffeff8 %ebpm ← %ebp
SFfact(4) — 0xbfffefe4 : 0x00000001 ← %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;
}
```

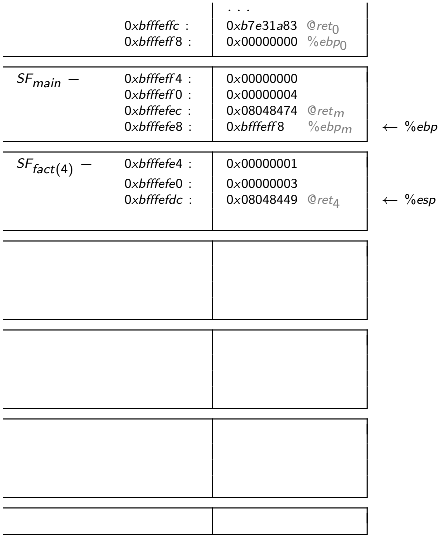


%eax

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;
}
```

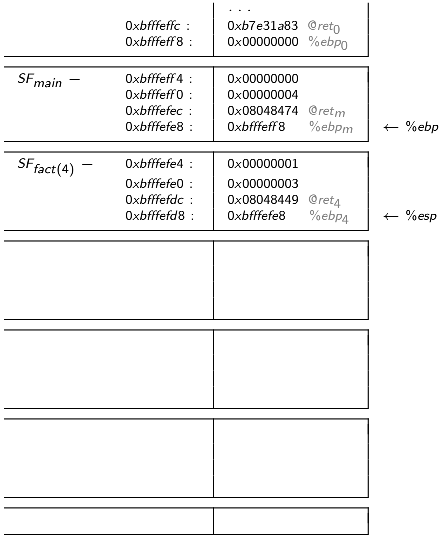
%eax



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	← %ebp
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	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;
}
```

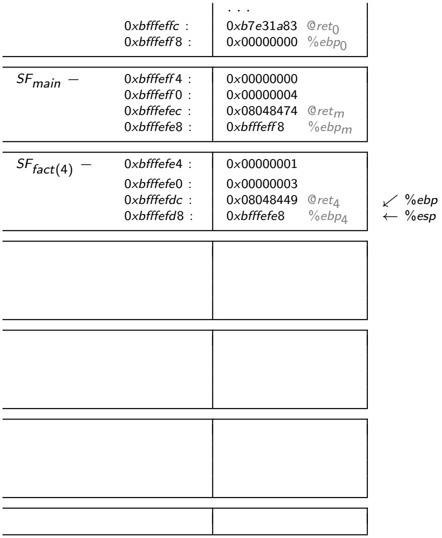


. . .
0xbfffeffc : 0xb7e31a83 @ret$_0$
0xbfffeff8 : 0x00000000 %ebp$_0$
SF_{main} —
0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @ret$_m$
0xbfffefe8 : 0xbfffeff8 %ebp$_m$
← %ebp
$SF_{fact(4)}$ —
0xbfffefe4 : 0x00000001
0xbfffefe0 : 0x00000003
0xbfffefdc : 0x08048449 @ret$_4$
0xbfffefd8 : 0xbfffefe8 %ebp$_4$
← %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;
}
```

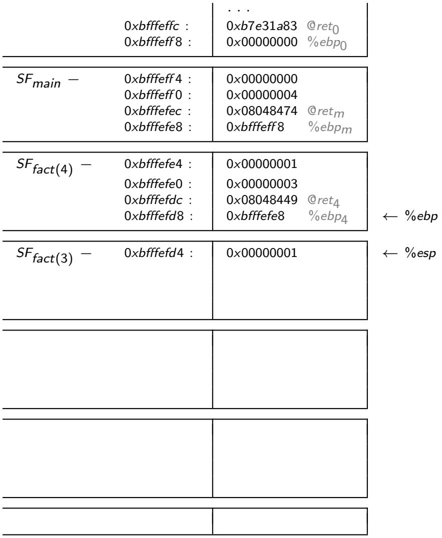


		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	↙ %ebp
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %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;
}
```

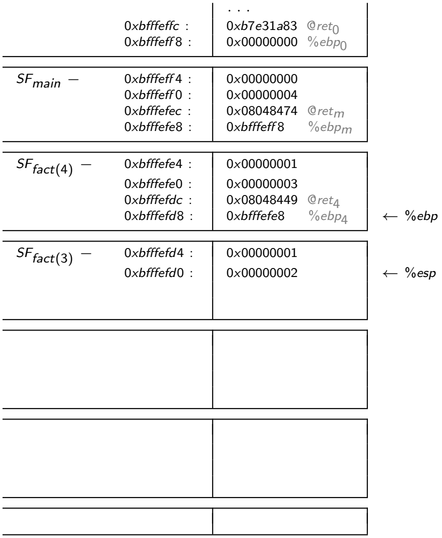


		. . .	
	0xbfffeffc :	0xb7e31a83	@ret_0
	0xbfffeff8 :	0x00000000	%ebp_0
SF_{main} —	0xbfffeff4 :	0x00000000	
	0xbfffeff0 :	0x00000004	
	0xbfffefec :	0x08048474	@ret_m
	0xbfffefe8 :	0xbfffeff8	%ebp_m
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001	
	0xbfffefe0 :	0x00000003	
	0xbfffefdc :	0x08048449	@ret_4
	0xbfffefd8 :	0xbfffefe8	%ebp_4
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001	

%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;
}
```

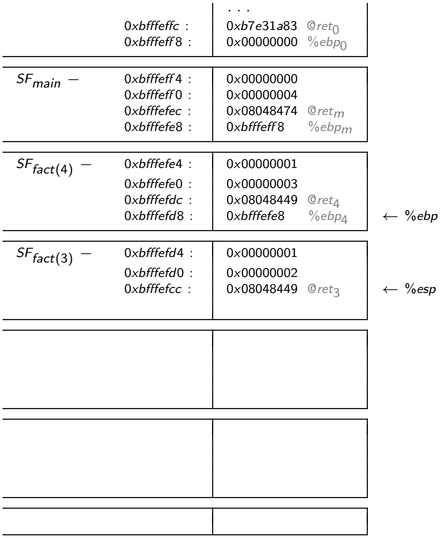


		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %ebp
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		← %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;
}
```

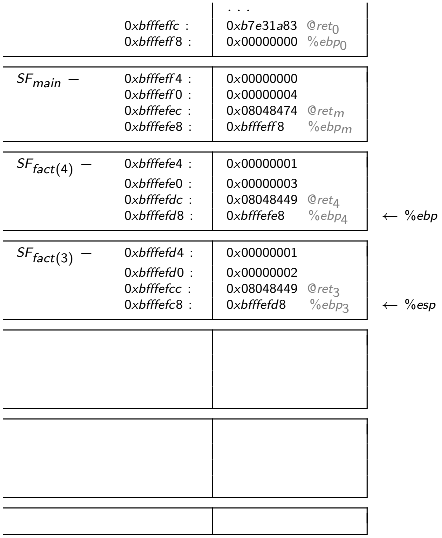


. . .
0xbfffeffc : 0xb7e31a83 @ret$_0$
0xbfffeff8 : 0x00000000 %ebp$_0$
SF_{main} –
0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @ret$_m$
0xbfffefe8 : 0xbfffeff8 %ebp$_m$
$SF_{fact(4)}$ –
0xbfffefe4 : 0x00000001
0xbfffefe0 : 0x00000003
0xbfffefdc : 0x08048449 @ret$_4$
0xbfffefd8 : 0xbfffefe8 %ebp$_4$
← %ebp
$SF_{fact(3)}$ –
0xbfffefd4 : 0x00000001
0xbfffefd0 : 0x00000002
0xbfffefcc : 0x08048449 @ret$_3$
← %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;
}
```

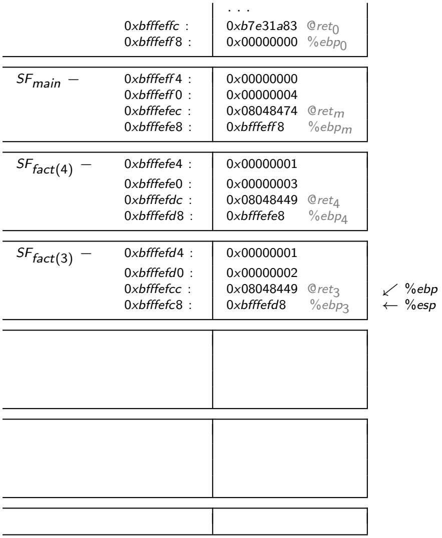


		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %ebp
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %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;
}
```



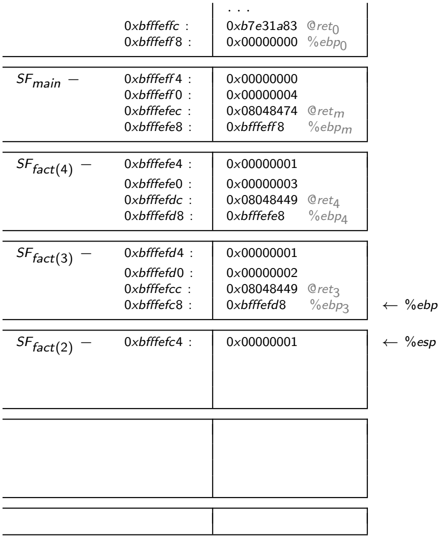
...
0xbfffeffc : 0xb7e31a83 @ret$_0$
0xbfffeff8 : 0x00000000 %ebp$_0$
SF_{main} —
0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @ret$_m$
0xbfffefe8 : 0xbfffeff8 %ebp$_m$
$SF_{fact(4)}$ —
0xbfffefe4 : 0x00000001
0xbfffefe0 : 0x00000003
0xbfffefdc : 0x08048449 @ret$_4$
0xbfffefd8 : 0xbfffefe8 %ebp$_4$
$SF_{fact(3)}$ —
0xbfffefd4 : 0x00000001
0xbfffefd0 : 0x00000002
0xbfffefcc : 0x08048449 @ret$_3$
0xbfffefc8 : 0xbfffefd8 %ebp$_3$
%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	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	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;
}
```

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		← %esp

%eax

x86 runtime memory - an example

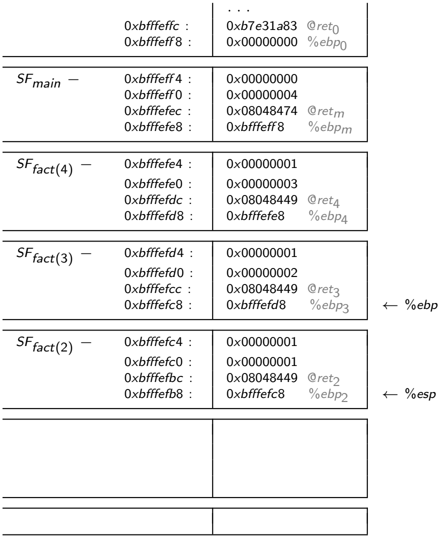
```
int fact(int x) {
  int y = 1;
  if (x == 0)
    return y;
  if (x > 0) {
    y = fact(x-1);
    return x*y;
  }
}

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

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	← %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;
}
```

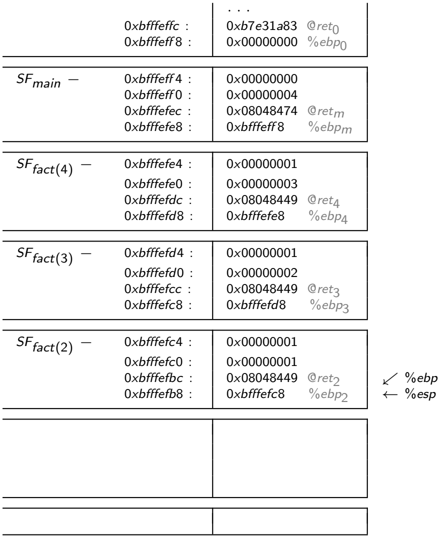


		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	← %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;
}
```



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret$_0$	
	0xbfffeff8 :	0x00000000	%ebp$_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret$_m$	
	0xbfffefe8 :	0xbfffeff8	%ebp$_m$	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret$_4$	
	0xbfffefd8 :	0xbfffefe8	%ebp$_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret$_3$	
	0xbfffefc8 :	0xbfffefd8	%ebp$_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret$_2$	$\swarrow$ %ebp
	0xbfffefb8 :	0xbfffefc8	%ebp$_2$	$\leftarrow$ %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;
}
```

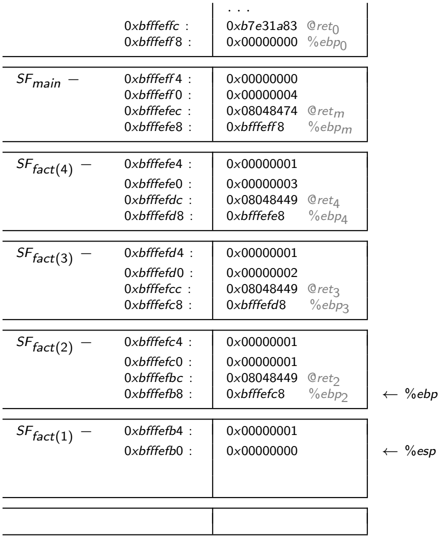
		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	← %ebp
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		← %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;
}
```

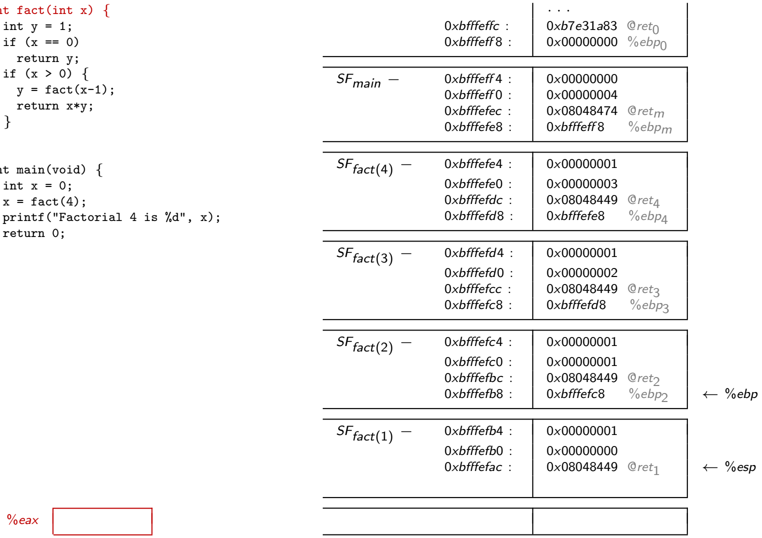
$\%eax$



		...		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	$\leftarrow \%ebp$
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		$\leftarrow \%esp$

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;
}
```



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	← %ebp
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	← %esp

x86 runtime memory - an example

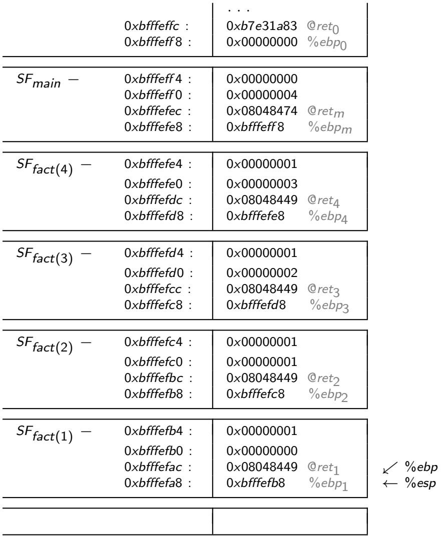
```
int fact(int x) {
  int y = 1;
  if (x == 0)
    return y;
  if (x > 0) {
    y = fact(x-1);
    return x*y;
  }
}

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

		. . .		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	← %ebp
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	$@ret_1$	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	← %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;
}
```



. . .
0xbfffeffc : 0xb7e31a83 @ret$_0$
0xbfffeff8 : 0x00000000 %ebp$_0$
SF$_{main}$ —
0xbfffeff4 : 0x00000000
0xbfffeff0 : 0x00000004
0xbfffefec : 0x08048474 @ret$_m$
0xbfffefe8 : 0xbfffeff8 %ebp$_m$
SF$_{fact(4)}$ —
0xbfffefe4 : 0x00000001
0xbfffefe0 : 0x00000003
0xbfffefdc : 0x08048449 @ret$_4$
0xbfffefd8 : 0xbfffefe8 %ebp$_4$
SF$_{fact(3)}$ —
0xbfffefd4 : 0x00000001
0xbfffefd0 : 0x00000002
0xbfffefcc : 0x08048449 @ret$_3$
0xbfffefc8 : 0xbfffefd8 %ebp$_3$
SF$_{fact(2)}$ —
0xbfffefc4 : 0x00000001
0xbfffefc0 : 0x00000001
0xbfffefbc : 0x08048449 @ret$_2$
0xbfffefb8 : 0xbfffefc8 %ebp$_2$
SF$_{fact(1)}$ —
0xbfffefb4 : 0x00000001
0xbfffefb0 : 0x00000000
0xbfffefac : 0x08048449 @ret$_1$
0xbfffefa8 : 0xbfffefb8 %ebp$_1$
%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;
}
```

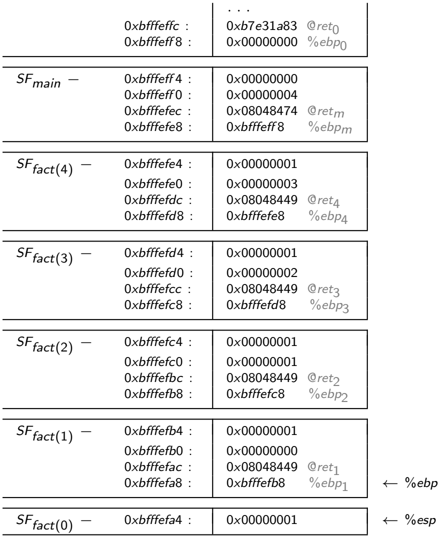
		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	← %ebp
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		← %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;
}
```

$\%eax$ 0x00000001



		. . .		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	$@ret_1$	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	$\leftarrow \%ebp$
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		$\leftarrow \%esp$

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;
}
```

$\%eax$ $\boxed{0x00000001}$

		. . .		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	$\leftarrow \%ebp$
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	$@ret_1$	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		$\leftarrow \%esp$

x86 runtime memory - an example

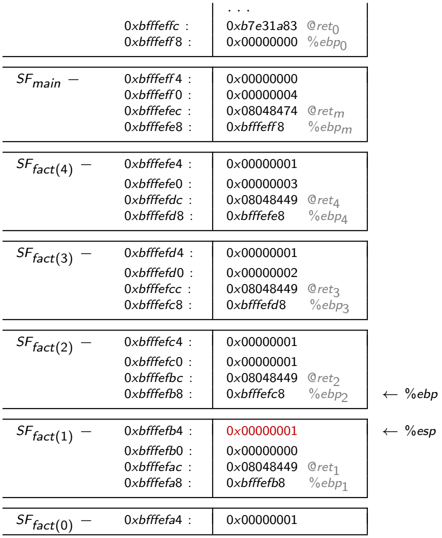
```c
int fact(int x) {
  int y = 1;
  if (x == 0)
    return y;
  if (x > 0) {
    y = fact(x-1);
    return x*y;
  }
}

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

$\%eax$ 0x00000001

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	$\leftarrow \%ebp$
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		$\leftarrow \%esp$
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	
$SF_{fact(0)}$ —	0xbfffefa4 :	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;
}
```



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} –	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ –	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ –	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ –	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	← %ebp
$SF_{fact(1)}$ –	0xbfffefb4 :	0x00000001		← %esp
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ –	0xbfffefa4 :	0x00000001		

%eax 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;
}
```

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	← %ebp
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		← %esp
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax 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 0x00000001

		. . .		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} –	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	
$SF_{fact(4)}$ –	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ –	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	$\leftarrow$ %ebp
$SF_{fact(2)}$ –	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	
$SF_{fact(1)}$ –	0xbfffefb4 :	0x00000001		$\leftarrow$ %esp
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	$@ret_1$	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	
$SF_{fact(0)}$ –	0xbfffefa4 :	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;
}
```

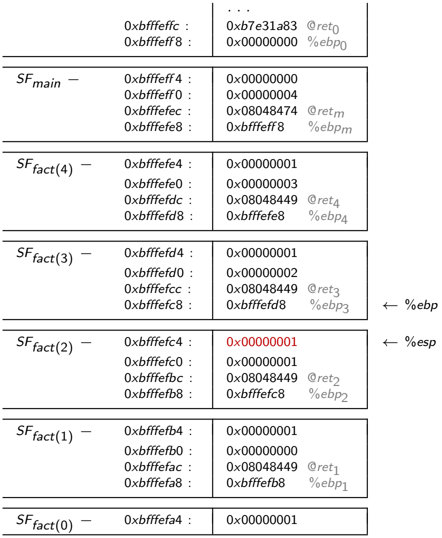
		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		← %esp
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax $\boxed{\text{0x00000001}}$

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 0x00000001



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		← %esp
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	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;
}
```

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	← %ebp
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		← %esp
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax 0x00000002

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;
}
```

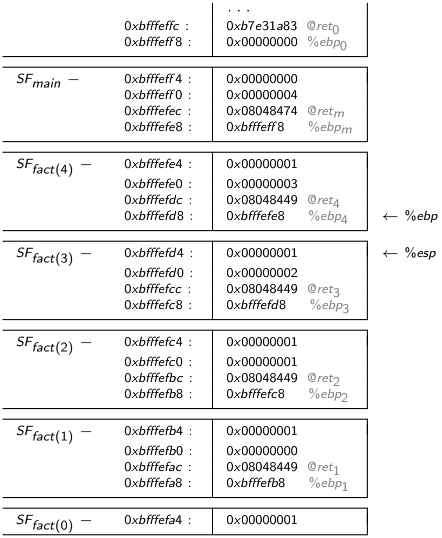
		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %ebp
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		← %esp
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax $\boxed{\text{0x00000002}}$

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

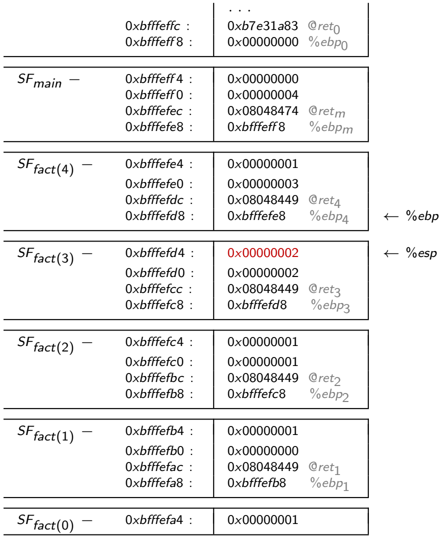


		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %*ebp*
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000001		← %*esp*
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	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_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %ebp
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		← %esp
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	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;
}
```

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	← %ebp
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		← %esp
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax $\boxed{0x00000006}$

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_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} –	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	← %ebp
$SF_{fact(4)}$ –	0xbfffefe4 :	0x00000001		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ –	0xbfffefd4 :	0x00000002		← %esp
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ –	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ –	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ –	0xbfffefa4 :	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;
}
```

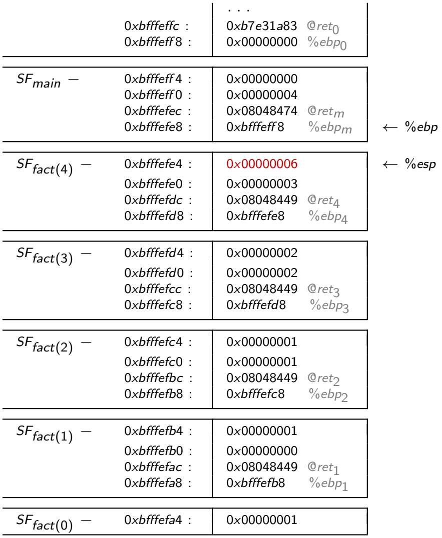
$\%eax$ 0x00000006

		...		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	$\leftarrow$ $\%ebp$
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000001		$\leftarrow$ $\%esp$
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	$@ret_1$	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	
$SF_{fact(0)}$ —	0xbfffefa4 :	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 $\boxed{0x00000006}$



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	
SF_{main} –	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	← %ebp
$SF_{fact(4)}$ –	0xbfffefe4 :	0x00000006		← %esp
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ –	0xbfffefd4 :	0x00000002		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ –	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ –	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ –	0xbfffefa4 :	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;
}
```

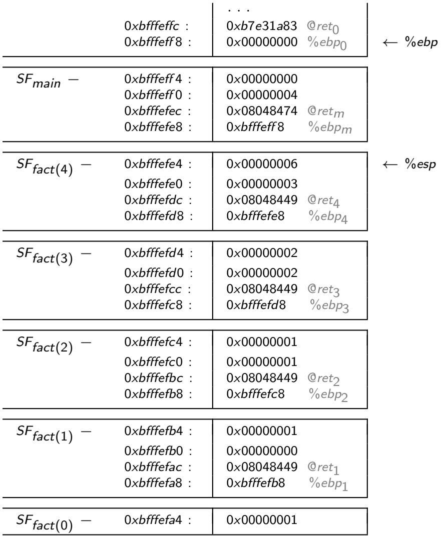
%eax 0x00000018

		. . .		
	0xbfffeffc :	0xb7e31a83	$@ret_0$	
	0xbfffeff8 :	0x00000000	$\%ebp_0$	
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	$@ret_m$	
	0xbfffefe8 :	0xbfffeff8	$\%ebp_m$	← %ebp
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000006		← %esp
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	$@ret_4$	
	0xbfffefd8 :	0xbfffefe8	$\%ebp_4$	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	$@ret_3$	
	0xbfffefc8 :	0xbfffefd8	$\%ebp_3$	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	$@ret_2$	
	0xbfffefb8 :	0xbfffefc8	$\%ebp_2$	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	$@ret_1$	
	0xbfffefa8 :	0xbfffefb8	$\%ebp_1$	
$SF_{fact(0)}$ —	0xbfffefa4 :	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 $\boxed{0x00000018}$



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	← %ebp
SF_{main} —	0xbfffeff4 :	0x00000000		
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000006		← %esp
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

x86 runtime memory - an example

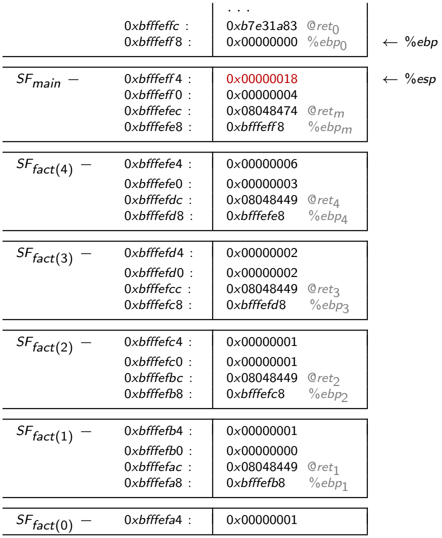
```
int fact(int x) {
  int y = 1;
  if (x == 0)
    return y;
  if (x > 0) {
    y = fact(x-1);
    return x*y;
  }
}

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

		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	← %ebp
SF_{main} —	0xbfffeff4 :	0x00000000		← %esp
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000006		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax $\boxed{\text{0x00000018}}$

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;
}
```



		. . .		
	0xbfffeffc :	0xb7e31a83	@ret_0	
	0xbfffeff8 :	0x00000000	%ebp_0	← %ebp
SF_{main} —	0xbfffeff4 :	0x00000018		← %esp
	0xbfffeff0 :	0x00000004		
	0xbfffefec :	0x08048474	@ret_m	
	0xbfffefe8 :	0xbfffeff8	%ebp_m	
$SF_{fact(4)}$ —	0xbfffefe4 :	0x00000006		
	0xbfffefe0 :	0x00000003		
	0xbfffefdc :	0x08048449	@ret_4	
	0xbfffefd8 :	0xbfffefe8	%ebp_4	
$SF_{fact(3)}$ —	0xbfffefd4 :	0x00000002		
	0xbfffefd0 :	0x00000002		
	0xbfffefcc :	0x08048449	@ret_3	
	0xbfffefc8 :	0xbfffefd8	%ebp_3	
$SF_{fact(2)}$ —	0xbfffefc4 :	0x00000001		
	0xbfffefc0 :	0x00000001		
	0xbfffefbc :	0x08048449	@ret_2	
	0xbfffefb8 :	0xbfffefc8	%ebp_2	
$SF_{fact(1)}$ —	0xbfffefb4 :	0x00000001		
	0xbfffefb0 :	0x00000000		
	0xbfffefac :	0x08048449	@ret_1	
	0xbfffefa8 :	0xbfffefb8	%ebp_1	
$SF_{fact(0)}$ —	0xbfffefa4 :	0x00000001		

%eax $\boxed{\texttt{0x00000018}}$

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)