```c
/* Вычисляем x + 3 */
int add3(int x) {
  int localx = x;
  incrk(&localx, 3);
  return localx;
}

/* увеличиваем значение на k */
void incrk(int *ip, int k) {
  *ip += k;
}
```

```asm
add3:
     push  ebp
     mov   ebp, esp
     sub   esp, 24   ; выделяем 24 байта
     mov   eax, dword [ebp + 8]
     mov   dword [ebp - 4], eax
       ; Присваиваем localx значение x
     mov dword [esp + 4], 3
       ; 2-ой параметр = 3
     lea eax, [ebp - 4]
       ; &localx
     mov dword [esp], eax
       ; 1-ый параметр = &localx
     call incrk
     mov eax, dword [ebp - 4]
       ; возвращаемое значение = localx
     leave
     ret
```

```c
typedef struct {
  int x;
  int y;
  int z;
} triple;

triple f2(int a, int b, int c) {
   triple v = {a, b, c};
   return v;
}

void f(triple *p) {
   *p = f2(1, 2, 3);
}
```

```asm
f:
  push    ebp                   ; (1)
  mov     ebp, esp              ; (2)
  sub     esp, 24               ; (3)
  mov     eax, 1                ; (4)
  mov     ecx, 3                ; (5)
  mov     edx, 2                ; (6)
  mov     dword [esp+4], eax    ; (7)
  mov     dword [esp+12], ecx   ; (8)
  mov     dword [esp+8], edx    ; (9)
  mov     eax, dword [ebp+8]    ; (10)
  mov     dword [esp], eax      ; (11)
  call    f2                    ; (12)
  sub     esp, 4                ; (13)
  leave                         ; (14)
  ret                           ; (15)
f2:
   push    ebp                  ; (1)
   mov     ebp, esp             ; (2)
   mov     eax, dword [ebp+8]   ; (3)
   mov     edx, dword [ebp+20]  ; (4)
   mov     dword [eax+8], edx   ; (5)
   mov     edx, dword [ebp+16]  ; (6)
   mov     dword [eax+4], edx   ; (7)
   mov     edx, dword [ebp+12]  ; (8)
   mov     dword [eax], edx     ; (9)
   pop     ebp                  ; (10)
   ret     4
```

```c
#include <stdio.h>

void nullify(int argc, char*
argv[]) {
}

int main(int argc, char* argv[])
{
   nullify(argc, argv);
   return 0;
}
```

```asm
CMAIN:
   lea     ecx, [esp+4]       ; (1)
   and     esp, -16           ; (2)
   push    dword [ecx-4]      ; (3)
   push    ebp                ; (4)
   mov     ebp, esp           ; (5)
   push    ecx                ; (6)
   sub     esp, 20            ; (7)
   mov     eax, dword [ecx+4] ; (8)
   mov     dword [esp+4], eax ; (9)
   mov     eax, dword [ecx]   ; (10)
   mov     dword [esp], eax   ; (11)
   call    nullify            ; (12)
   mov     eax, 0             ; (13)
   add     esp, 20            ; (14)
   pop     ecx                ; (15)
   pop     ebp                ; (16)
   lea     esp, [ecx-4]       ; (17)
   ret

nullify:
        ret
```

Возвращаемое значение - структура

```c
#include <stdlib.h>

struct chain;

typedef struct chain {
   int val;
   struct chain *next;
} t_chain, *p_chain;

p_chain insert(p_chain p, int val) {
   if ((0 == p) || (p->val > val)) {
      p_chain np = (p_chain)malloc(sizeof(t_chain));
      np->val = val;
      np->next = p;
      return np;
   } else {
      p->next = insert(p, val);
      return p;
   }
}
```

```nasm
%include 'io.inc'

section .text

CEXTERN malloc

insert:
   push    ebp
   mov     ebp, esp
   sub     esp, 24

   mov     dword [ebp-4], esi
   mov     esi, dword [ebp+8]
   mov     dword [ebp-8], ebx
   mov     ebx, dword [ebp+12]

   test    esi, esi
   je      .L2
   cmp     dword [esi], ebx
   jle     .L3
.L2:
   mov     dword [esp], 8
   call    malloc
   mov     dword [eax], ebx
   mov     dword [eax+4], esi
   mov     ebx, dword [ebp-8]
   mov     esi, dword [ebp-4]
   mov     esp, ebp
   pop     ebp
   ret
.L3:
   mov     dword [esp+4], ebx
   mov     dword [esp], esi
   call    insert
   mov     dword [esi+4], eax
   mov     eax, esi
   mov     ebx, dword [ebp-8]
   mov     esi, dword [ebp-4]
   mov     esp, ebp
   pop     ebp
   ret
```

Соглашение о вызовах - stdcall

```c
#include <stdio.h>

__attribute__((stdcall))
int sum(int x, int y);

int main() {
   int a = 1, b = 2, c;
   c = sum(a, b);
   printf("%d\n", c);
   return 0;
}

__attribute__((stdcall))
int sum(int x, int y) {
   int t = x + y;
   return t;
}
```

```asm
sum:
   push    ebp
   mov     ebp, esp
   sub     esp, 16
   mov     edx, dword [ebp+12]
   mov     eax, dword [ebp+8]
   add     eax, edx
   mov     dword [ebp-4], eax
   mov     eax, dword [ebp-4]
   leave
   ret     8

CMAIN:
   ; …
   mov     eax, dword [ebp-12]
   mov     dword [esp+4], eax
   mov     eax, dword [ebp-16]
   mov     dword [esp], eax
   call    sum
   sub     esp, 8
   mov     dword [ebp-8], eax
   ; …
```

Соглашение о вызовах - fastcall

```c
#include <stdio.h>

__attribute__((fastcall)) int
sum(int x, int y);

int main() {
   int a = 1, b = 2, c;
   c = sum(a, b);
   printf("%d\n", c);
   return 0;
}

__attribute__((fastcall)) int
sum(int x, int y) {
   int t = x + y;
   return t;
}
```

```asm
CMAIN:
   ; …
   mov     edx, DWORD [ebp-12]
   mov     ecx, DWORD [ebp-16]
   call    sum
   mov     DWORD [ebp-8], eax
   ; …

sum:
   lea     eax, [ecx + edx]
   ret
```

Отказ от указателя фрейма

```c
void f(int x, int y) {
   int numerator =
      (x + y) * (x - y);
   int denominator =
      x * x + y * y;
   if (0 == denominator) {
      denominator = 1;
   }
   return (100 * numerator) /
            denominator;
}
```

```asm
;Сохраняемые регистры:
; esi = [esp + 4]
; ebx = [esp]

;Параметры:
; esi = y
; ecx = x
; ebx = x^2 + y^2
; edx = (x + y) * (x - y)

f:
   ; пролог
   sub     esp, 8
   mov     dword [esp+4], esi
   mov     esi, dword [esp+16]
   mov     ecx, dword [esp+12]
   mov     dword [esp], ebx

   mov     edx, esi
   imul    edx, esi ; edx = y^2
   mov     eax, ecx
   imul    eax, ecx ; eax = x^2
   mov     ebx, edx
   add     ebx, eax
             ; ebx = x^2 + y^2
   jne     .L2
   mov     ebx, 1
.L2
   lea     edx, [esi+ecx]
   sub     ecx, esi
   imul    edx, ecx
   imul    edx, edx, 100
   mov     eax, edx
   sar     edx, 31
   idiv    ebx
   mov     esi, DWORD [esp+4]
   mov     ebx, DWORD [esp]
   add     esp, 8
   ret
```

Переменное количество параметров

```
#include <stdarg.h>

int average(int count, ...) {
    va_list ap;
    int sum = 0;
    if (0 == count) {
       return -1;
    }
    va_start(ap, count);
    for (int i=0; i<count; i++)
        sum += va_arg(ap, int);
    va_end(ap);
    return sum/count;
}
```
	```
	average:
	  push ebp
	  mov  ebp, esp
	  push ebx
	  mov  ecx, dword [ebp+8]
	  test ecx, ecx
	  jne  .L11
	  mov  eax, -1
	  pop  ebx
	  pop  ebp
	  ret
	.L11:
	  xor  eax, eax
	  xor  edx, edx
	  test ecx, ecx
	  lea  ebx, [ebp+12]
	  jle  .L5
	.L8:
	  add  edx, dword [ebx+eax*4]
	  add  eax, 1
	  cmp  ecx, eax
	  jg   .L8
	.L5:
	  mov  eax, edx
	  sar  edx, 31
	  idiv ecx
	  pop  ebx
	  pop  ebp
	  ret
	```