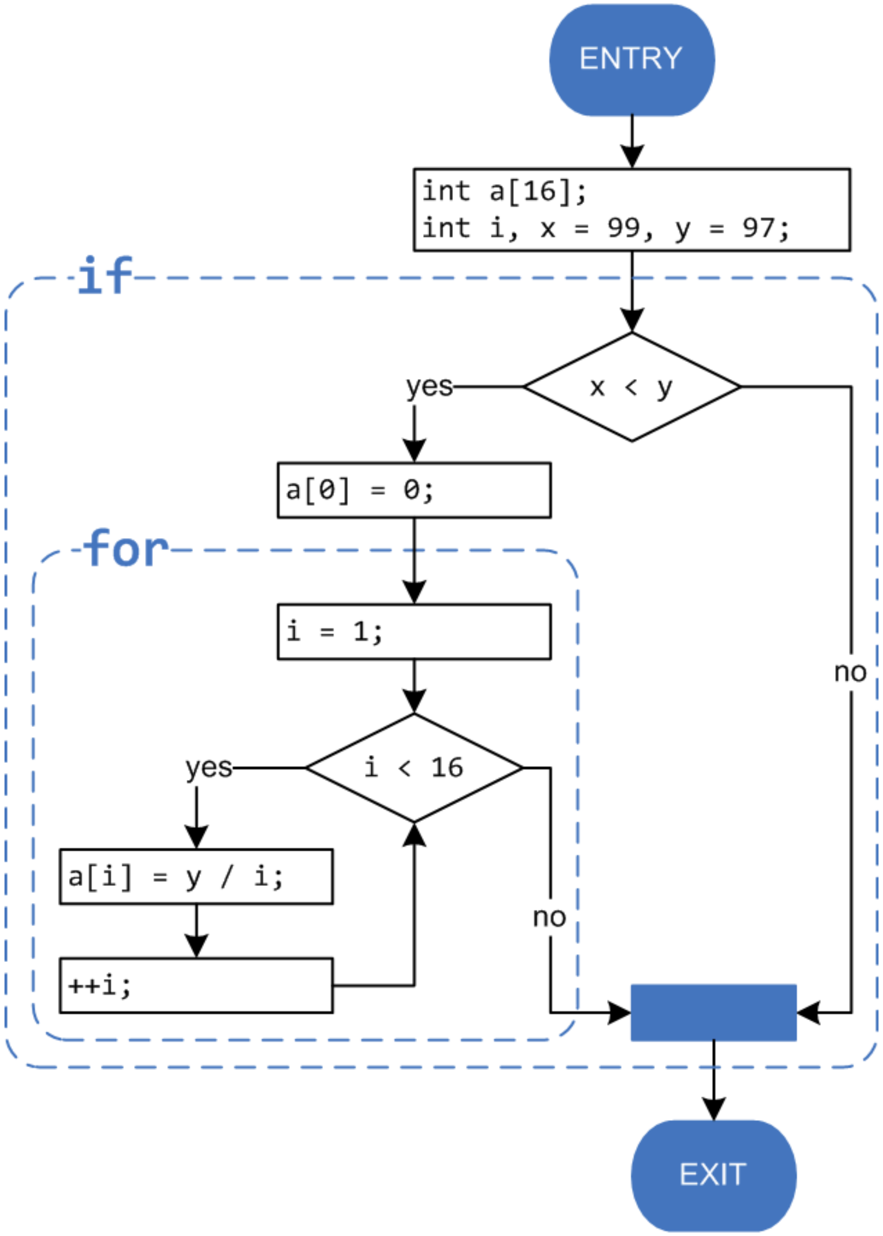
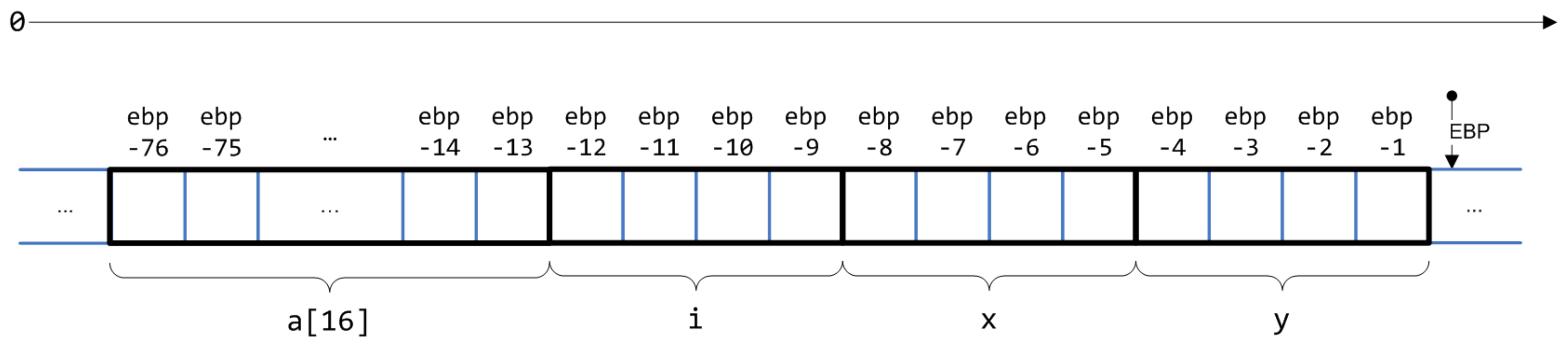
Лекция 4

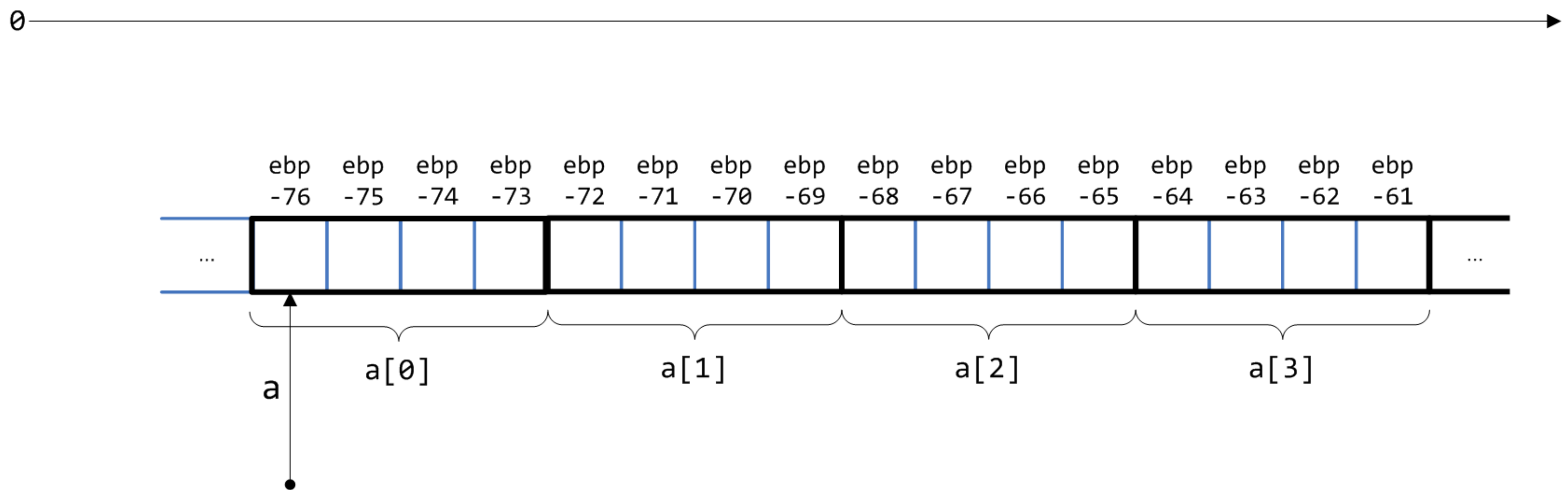
19 февраля

```
void f() {
   int a[16];
   int i, x = 99, y = 97;          // 1
   if (x < y) {                    // 2
      a[0] = 0;                    // 3
      for (i = 1; i < 16; ++i) {   // 4
         a[i] = y / i;             // 5
      }
   }
}
```

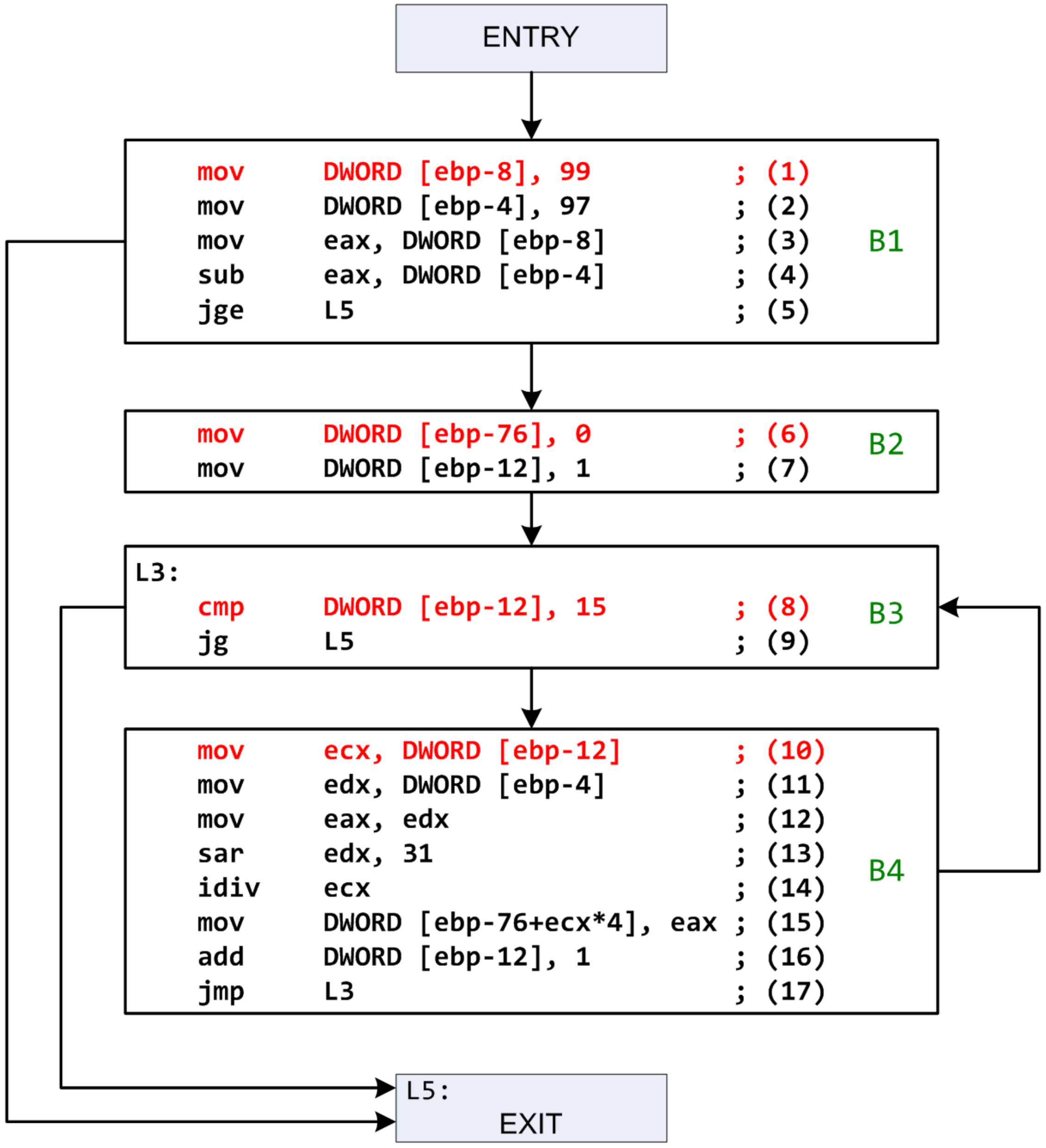
```
section .text
global f
f:
    push    ebp
    mov     ebp, esp
    sub     esp, 88
    mov     DWORD [ebp-8], 99              ; (1)
    mov     DWORD [ebp-4], 97              ; (2)
    mov     eax, DWORD [ebp-8]             ; (3)
    sub     eax, DWORD [ebp-4]             ; (4)
    jge     L5                             ; (5)
    mov     DWORD [ebp-76], 0              ; (6)
    mov     DWORD [ebp-12], 1              ; (7)
L3:
    cmp     DWORD [ebp-12], 15             ; (8)
    jg      L5                             ; (9)
    mov     ecx, DWORD [ebp-12]            ; (10)
    mov     edx, DWORD [ebp-4]             ; (11)
    mov     eax, edx                       ; (12)
    sar     edx, 31                        ; (13)
    idiv    ecx                            ; (14)
    mov     DWORD [ebp-76+ecx*4], eax      ; (15)
    add     DWORD [ebp-12], 1              ; (16)
    jmp     L3                             ; (17)
L5:
    leave
    ret
```


ENTRY
int a[16];
int i, x = 99, y = 97;
if
x < y
yes
no
a[0] = 0;
for
i = 1;
i < 16
yes
no
a[i] = y / i;
++i;
EXIT


0
ebp -76
ebp -75
…
ebp -14
ebp -13
ebp -12
ebp -11
ebp -10
ebp -9
ebp -8
ebp -7
ebp -6
ebp -5
ebp -4
ebp -3
ebp -2
ebp -1
EBP
…
…
…
a[16]
i
x
y


0
ebp -76
ebp -75
ebp -74
ebp -73
ebp -72
ebp -71
ebp -70
ebp -69
ebp -68
ebp -67
ebp -66
ebp -65
ebp -64
ebp -63
ebp -62
ebp -61
...
...
a[0]
a[1]
a[2]
a[3]
a

```
        mov     DWORD [ebp-8], 99             ; (1)
        mov     DWORD [ebp-4], 97             ; (2)
        mov     eax, DWORD [ebp-8]            ; (3)
        sub     eax, DWORD [ebp-4]            ; (4)
        jge     L5                            ; (5)
        mov     DWORD [ebp-76], 0             ; (6)
        mov     DWORD [ebp-12], 1             ; (7)
    L3:
        cmp     DWORD [ebp-12], 15            ; (8)
        jg      L5                            ; (9)
        mov     ecx, DWORD [ebp-12]           ; (10)
        mov     edx, DWORD [ebp-4]            ; (11)
        mov     eax, edx                      ; (12)
        sar     edx, 31                       ; (13)
        idiv    ecx                           ; (14)
        mov     DWORD [ebp-76+ecx*4], eax     ; (15)
        add     DWORD [ebp-12], 1             ; (16)
        jmp     L3                            ; (17)
    L5:
        …
```



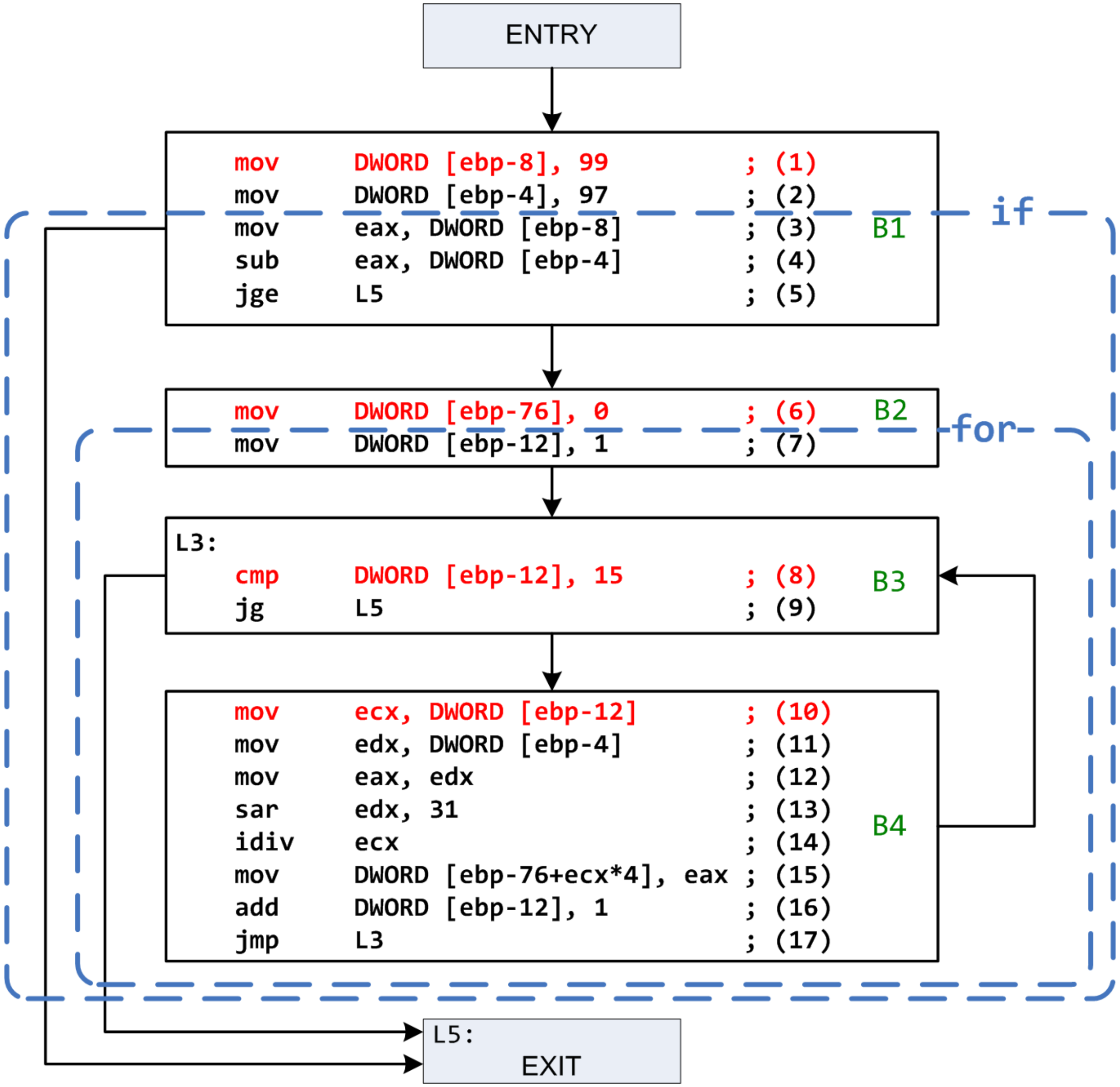
```
    mov     DWORD [ebp-8], 99           ; (1)
    mov     DWORD [ebp-4], 97           ; (2)
    mov     eax, DWORD [ebp-8]          ; (3)
    sub     eax, DWORD [ebp-4]          ; (4)
    jge     L5                          ; (5)

    mov     DWORD [ebp-76], 0           ; (6)
    mov     DWORD [ebp-12], 1           ; (7)

L3:
    cmp     DWORD [ebp-12], 15          ; (8)
    jg      L5                          ; (9)

    mov     ecx, DWORD [ebp-12]         ; (10)
    mov     edx, DWORD [ebp-4]          ; (11)
    mov     eax, edx                    ; (12)
    sar     edx, 31                     ; (13)
    idiv    ecx                         ; (14)
    mov     DWORD [ebp-76+ecx*4], eax ; (15)
    add     DWORD [ebp-12], 1           ; (16)
    jmp     L3                          ; (17)

L5:
```

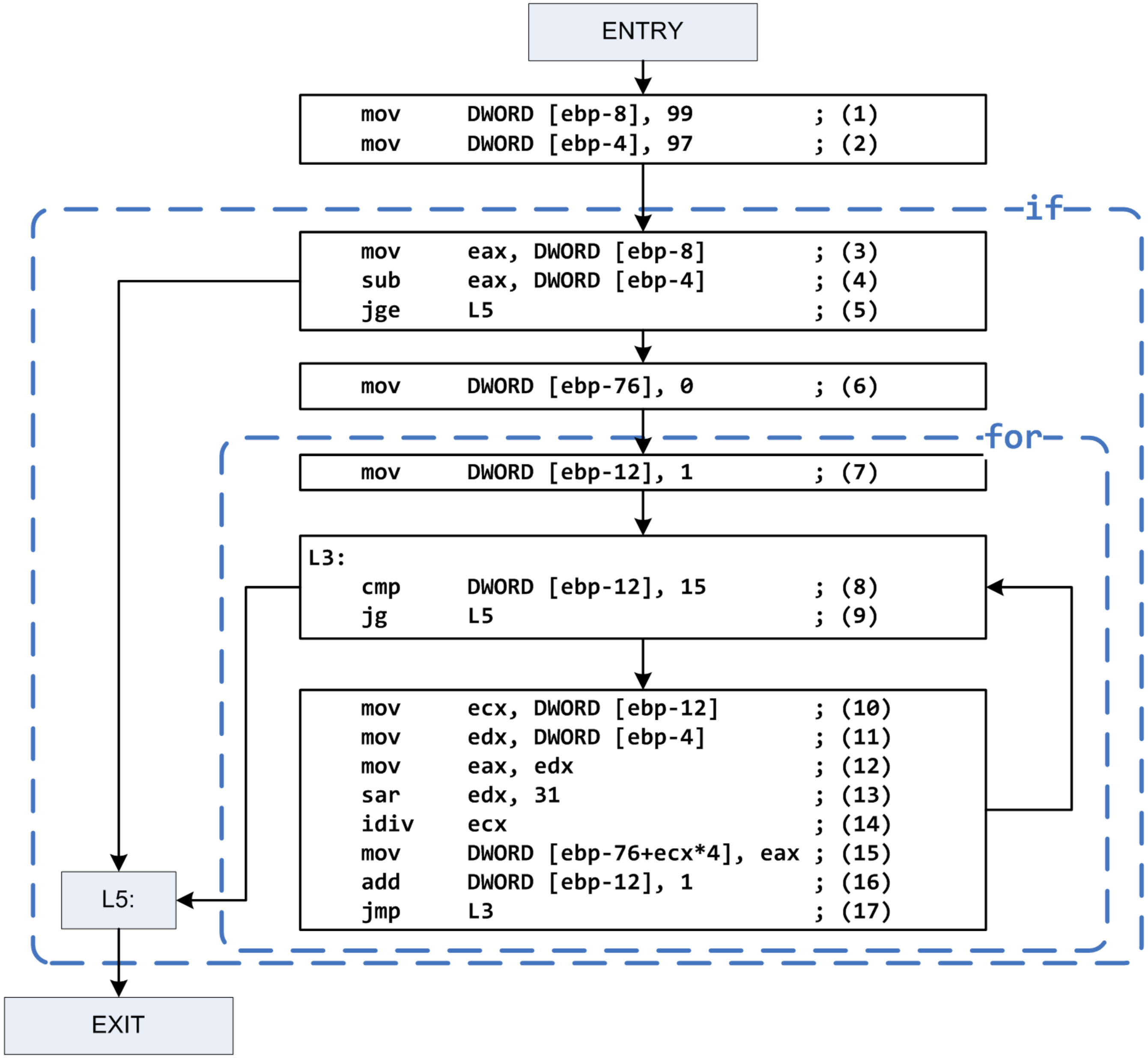


```
    mov     DWORD [ebp-8], 99         ; (1)
    mov     DWORD [ebp-4], 97         ; (2)
    mov     eax, DWORD [ebp-8]        ; (3)
    sub     eax, DWORD [ebp-4]        ; (4)
    jge     L5                        ; (5)

    mov     DWORD [ebp-76], 0         ; (6)
    mov     DWORD [ebp-12], 1         ; (7)

L3:
    cmp     DWORD [ebp-12], 15        ; (8)
    jg      L5                        ; (9)

    mov     ecx, DWORD [ebp-12]       ; (10)
    mov     edx, DWORD [ebp-4]        ; (11)
    mov     eax, edx                  ; (12)
    sar     edx, 31                   ; (13)
    idiv    ecx                       ; (14)
    mov     DWORD [ebp-76+ecx*4], eax ; (15)
    add     DWORD [ebp-12], 1         ; (16)
    jmp     L3                        ; (17)
```



```
    mov     DWORD [ebp-8], 99           ; (1)
    mov     DWORD [ebp-4], 97           ; (2)
```

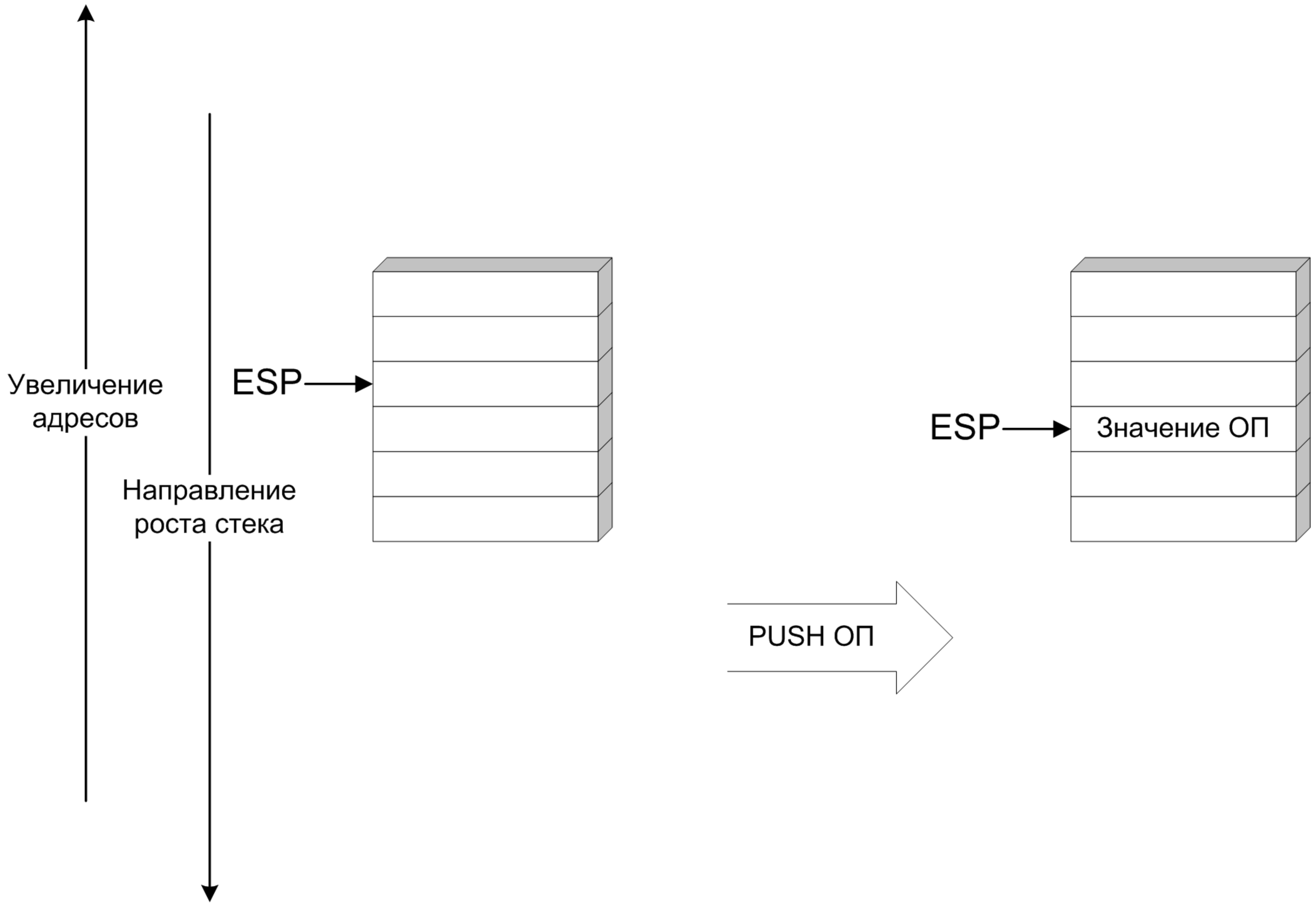
```
    mov     eax, DWORD [ebp-8]          ; (3)
    sub     eax, DWORD [ebp-4]          ; (4)
    jge     L5                          ; (5)
```

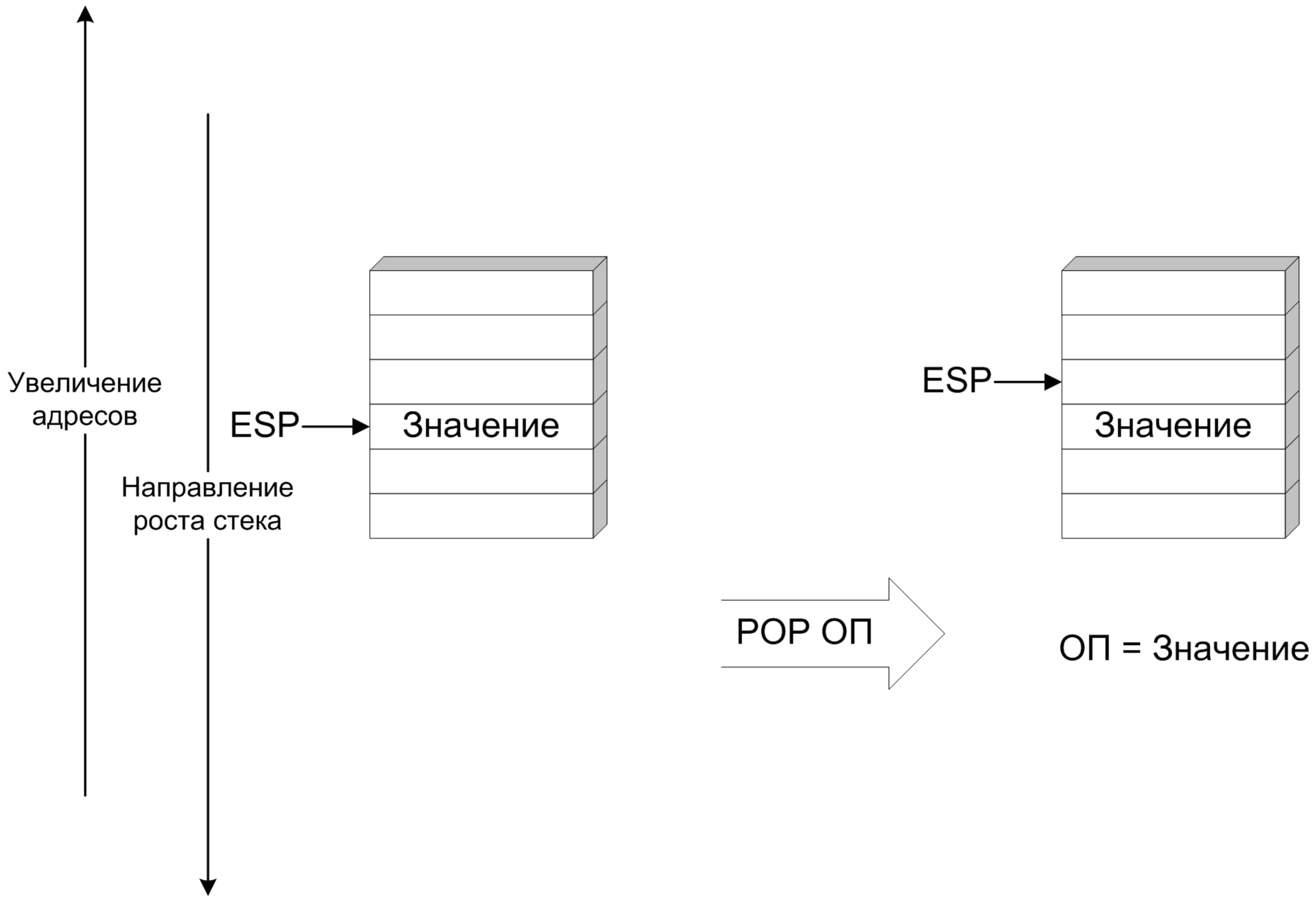
```
    mov     DWORD [ebp-76], 0           ; (6)
```

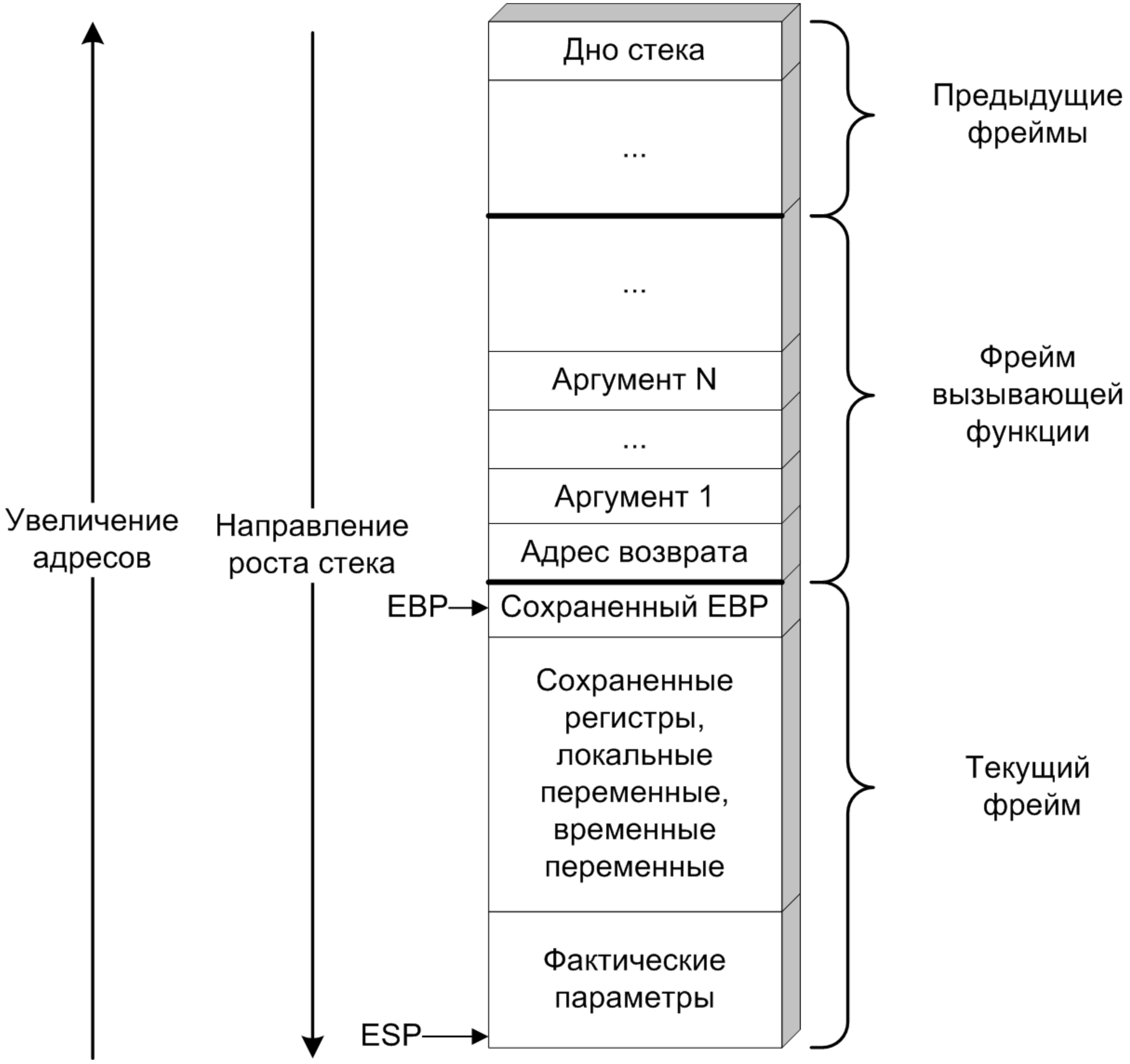
```
    mov     DWORD [ebp-12], 1           ; (7)
```

```
L3:
    cmp     DWORD [ebp-12], 15          ; (8)
    jg      L5                          ; (9)
```

```
    mov     ecx, DWORD [ebp-12]         ; (10)
    mov     edx, DWORD [ebp-4]          ; (11)
    mov     eax, edx                    ; (12)
    sar     edx, 31                     ; (13)
    idiv    ecx                         ; (14)
    mov     DWORD [ebp-76+ecx*4], eax ; (15)
    add     DWORD [ebp-12], 1           ; (16)
    jmp     L3                          ; (17)
```


Увеличение адресов
Направление роста стека
ESP
PUSH ОП
ESP
Значение ОП


Увеличение адресов
Направление роста стека
ESP
Значение
POP ОП
ESP
Значение
ОП = Значение


Увеличение адресов
Направление роста стека
Дно стека
...
...
Аргумент N
...
Аргумент 1
Адрес возврата
EBP
Сохраненный EBP
Сохраненные регистры, локальные переменные, временные переменные
Фактические параметры
ESP
Предыдущие фреймы
Фрейм вызывающей функции
Текущий фрейм

```c
int main() {
   int a = 1, b = 2, c;
   c = sum(a, b);
   return 0;
}

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

```nasm
%include ‘io.inc’
section .text

global CMAIN
CMAIN:
mov    DWORD [ebp-16],0x1  ; (1)
mov    DWORD [ebp-12],0x2  ; (2)
mov    eax,DWORD [ebp-12]  ; (3)
mov    DWORD [esp+4],eax   ; (4)
mov    eax,DWORD [ebp-16]  ; (5)
mov    DWORD [esp],eax     ; (6)
call   sum                 ; (7)
mov    DWORD [ebp-8],eax   ; (8)

global sum
sum:
push   ebp                 ; (9)
mov    ebp,esp             ; (10)
sub    esp,0x10            ; (11)
mov    edx,DWORD [ebp+12]  ; (12)
mov    eax,DWORD [ebp+8]   ; (13)
add    eax,edx             ; (14)
mov    DWORD [ebp-4],eax   ; (15)
mov    eax,DWORD [ebp-4]   ; (16)
mov    esp, ebp            ; (17)
pop    ebp                 ; (18)
ret                        ; (19)
```