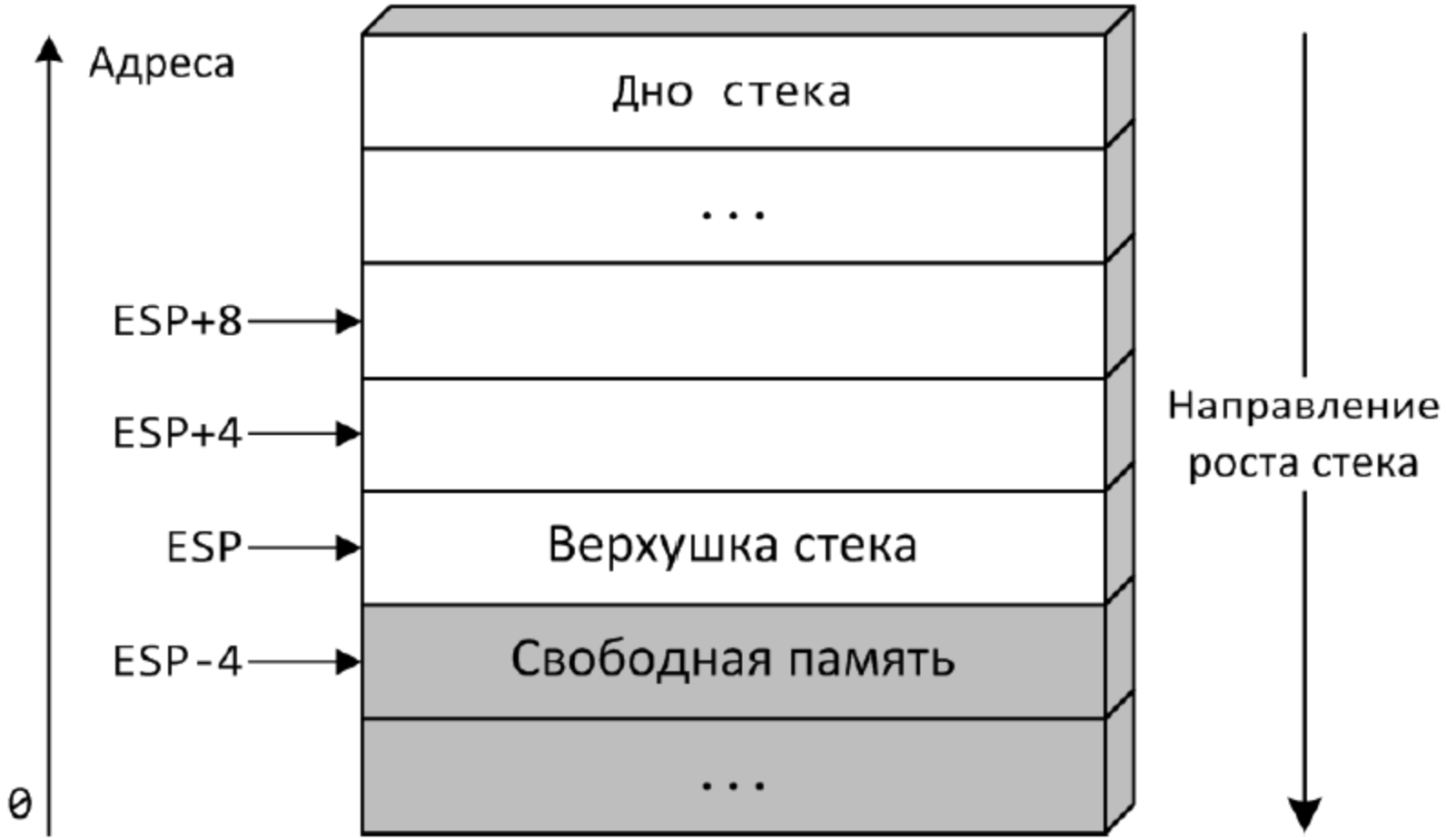
Семинар 19.03.2021

Аппаратный стек



push/pop

`push r/m/imm 32` - положить значение в стек

`pop r/m 32` - забрать значение из стека

```
push eax
```

эквивалентно

```
sub esp, 4
mov dword[esp], eax
```

```
pop eax
```

эквивалентно

```
mov eax, dword[esp]
add esp, 4
```

call/ret

`call r/m/label 32` - вызвать функцию

`ret` - вернуться из функции

`ret imm16` - вернуться из функции с очисткой стека

```
call foo
```

эквивалентно

```
push <адрес следующей инструкции (адрес возврата)>
jmp foo
```

```
ret
```

эквивалентно

```
pop <reg>
jmp <reg>
```

```
ret 4
```

эквивалентно

```
pop <reg>
add esp, 4
jmp <reg>
```

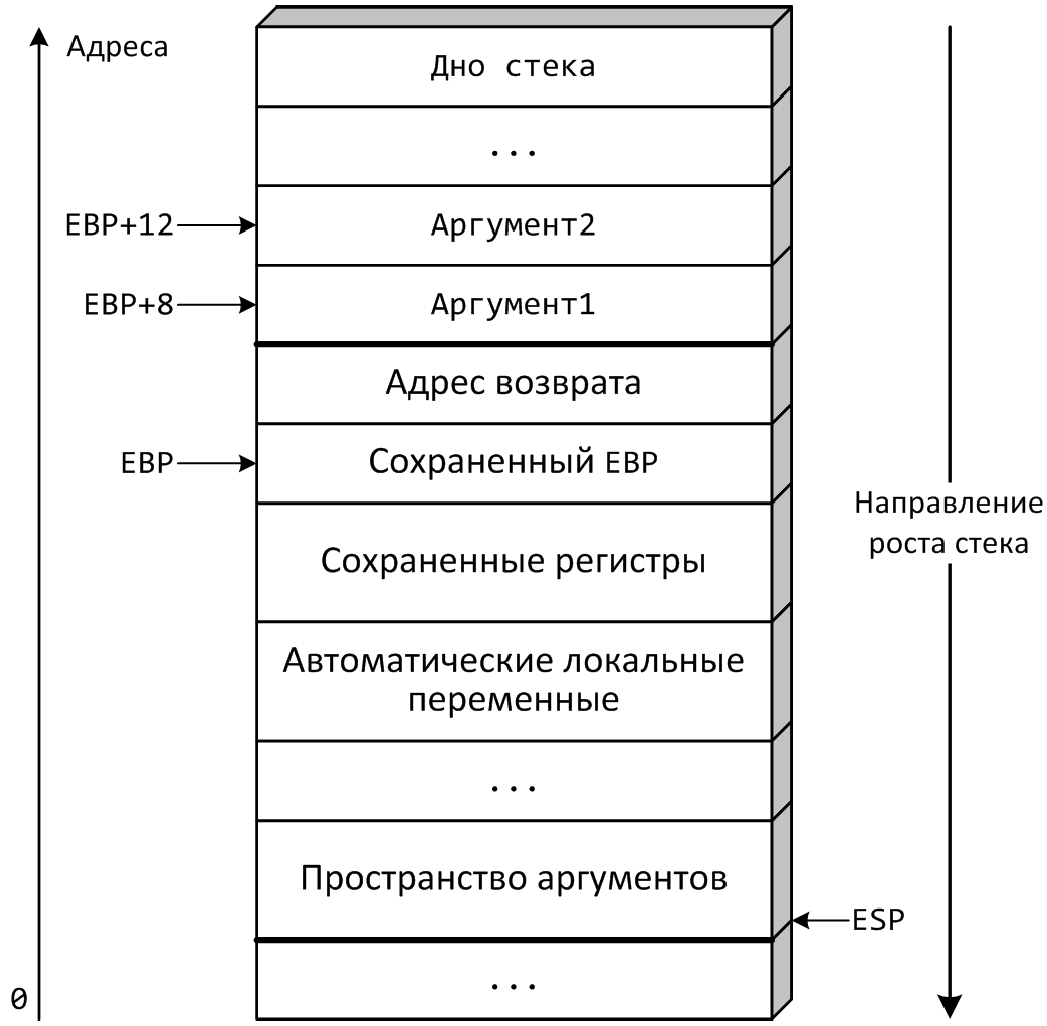
Соглашение о вызове (calling convention)

Это описание правил вызова функций:

1. Как вызываются функции
2. Как происходит возврат из функций
3. Как передаются аргументы
4. Как возвращаются значения
5. Как используются регистры

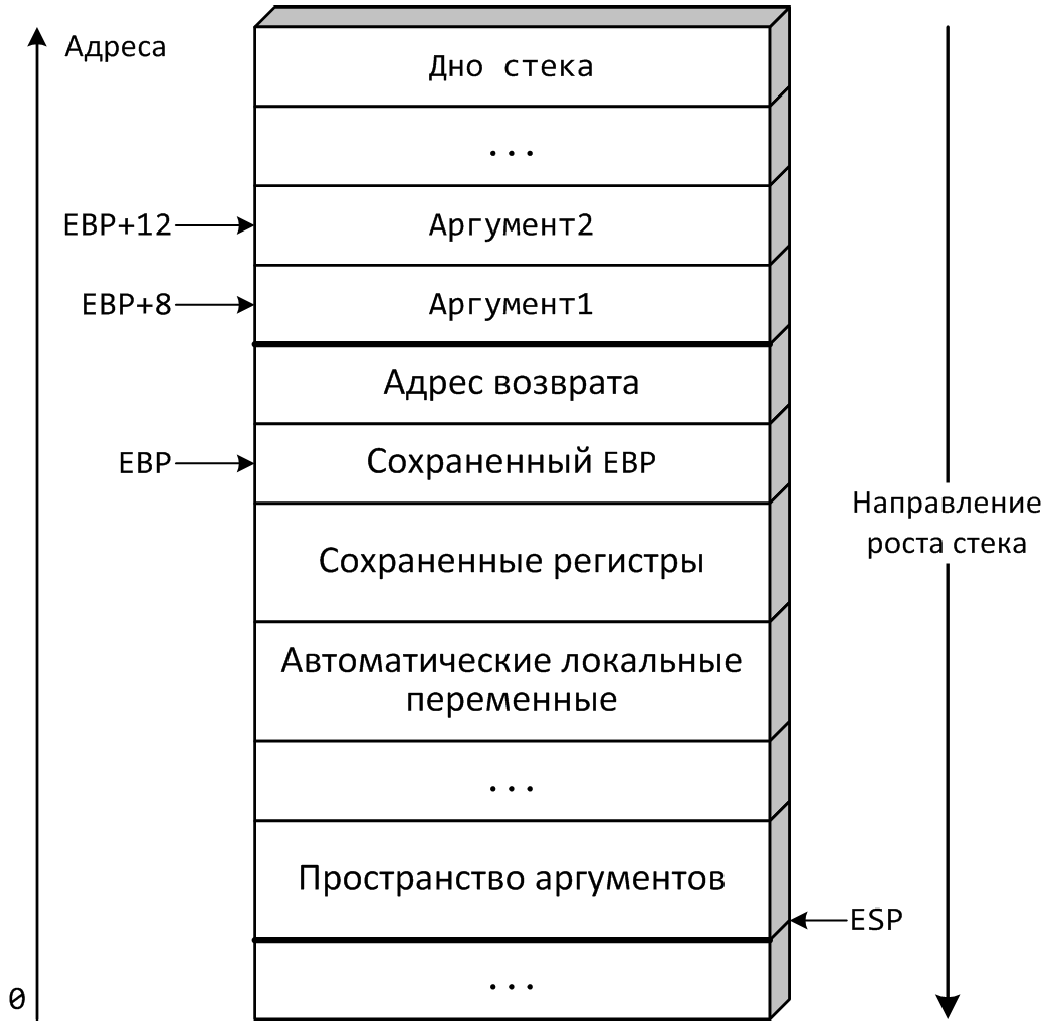
Соглашение о вызове cdecl

- Вызов через call
- Возврат через ret
- Передача аргументов через стек двойными словами в обратном порядке (на вершине первый аргумент)



Соглашение о вызове cdecl

- Возвращаемое значение в al/ax/eax/edx:eax
- Локальные переменные на стеке (мусор)



Соглашение о вызове cdecl

- Пролог (в начале)

```
push ebp
mov ebp, esp
```

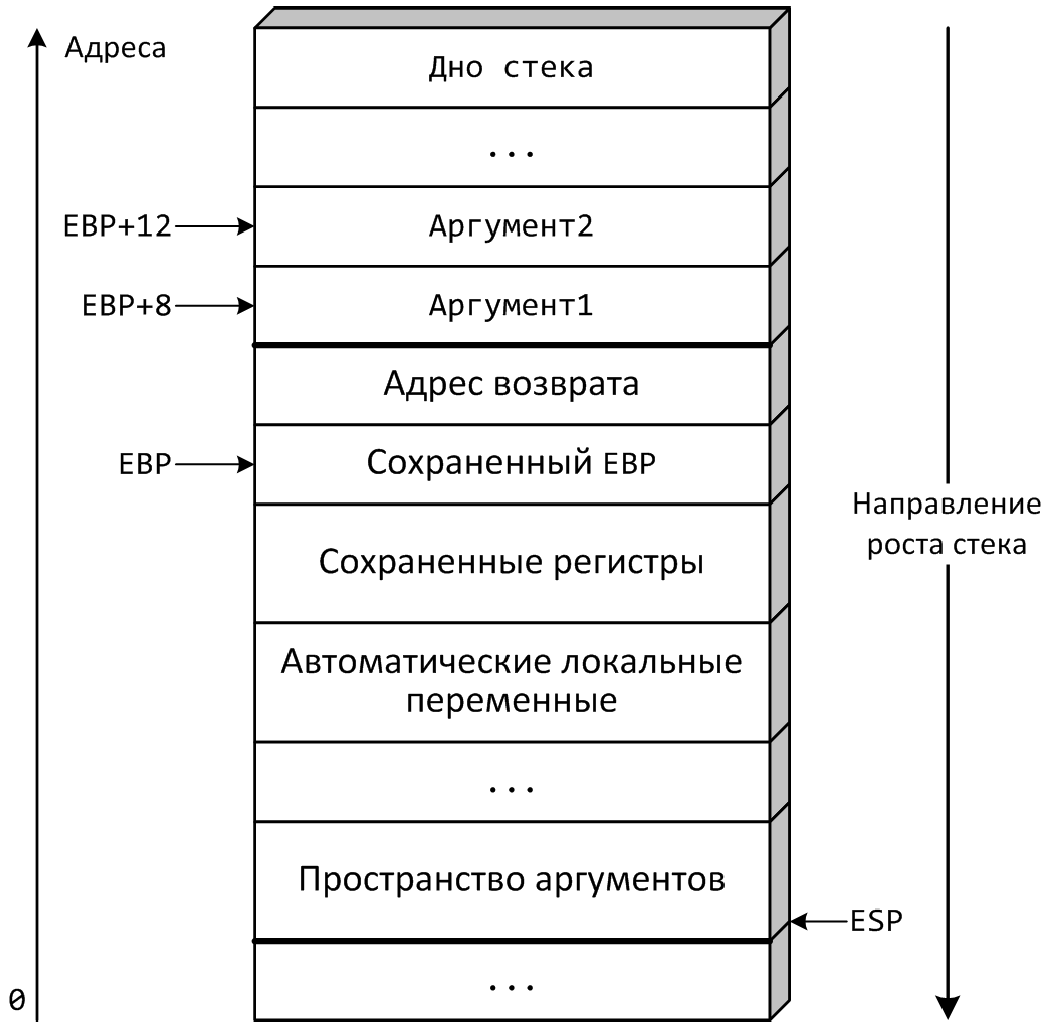
- Эпилог (в конце)

```
mov esp, ebp
pop ebp
```

или

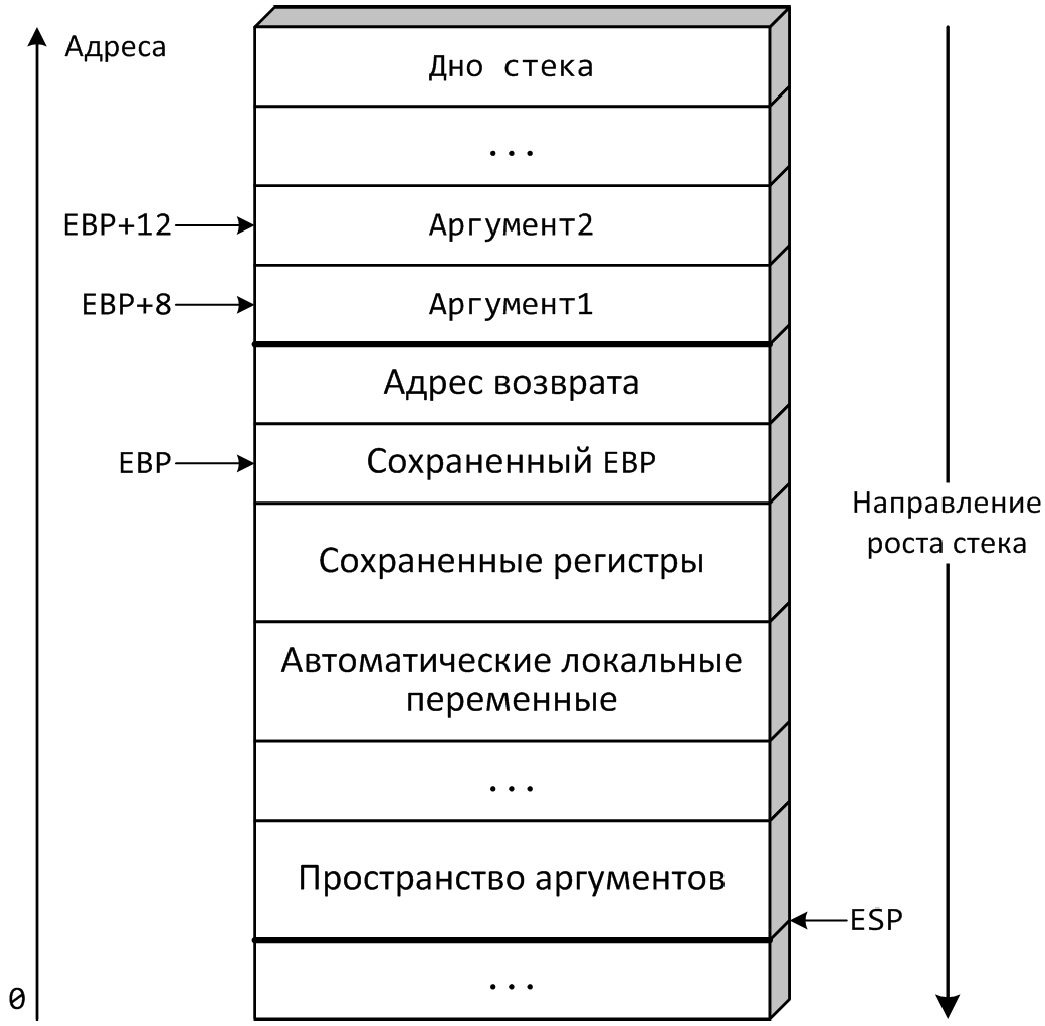
```
leave
```

- Доступ к параметрам и локальным переменным через ebp



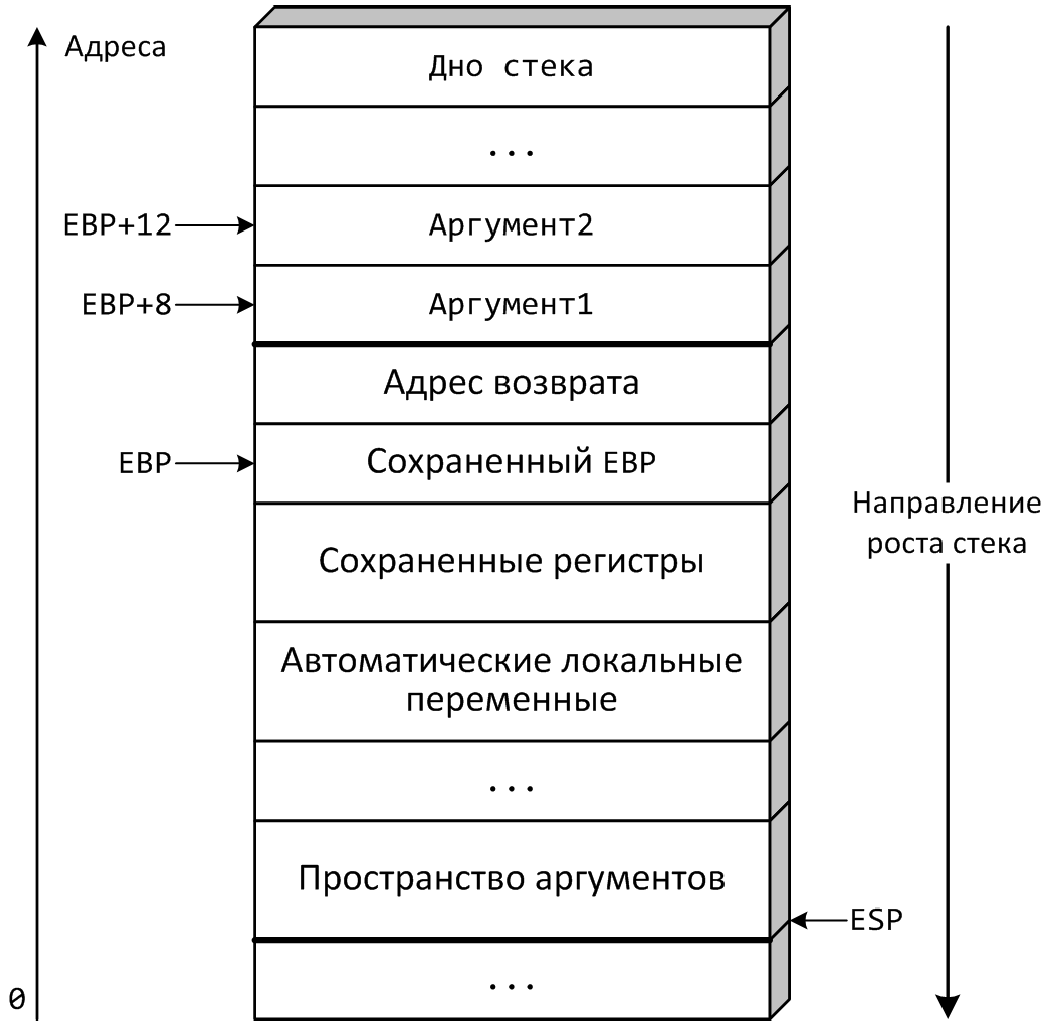
Соглашение о вызове cdecl

- Фрейм
- Очистка стека от параметров в вызывающей функции (`ret` без операнда)



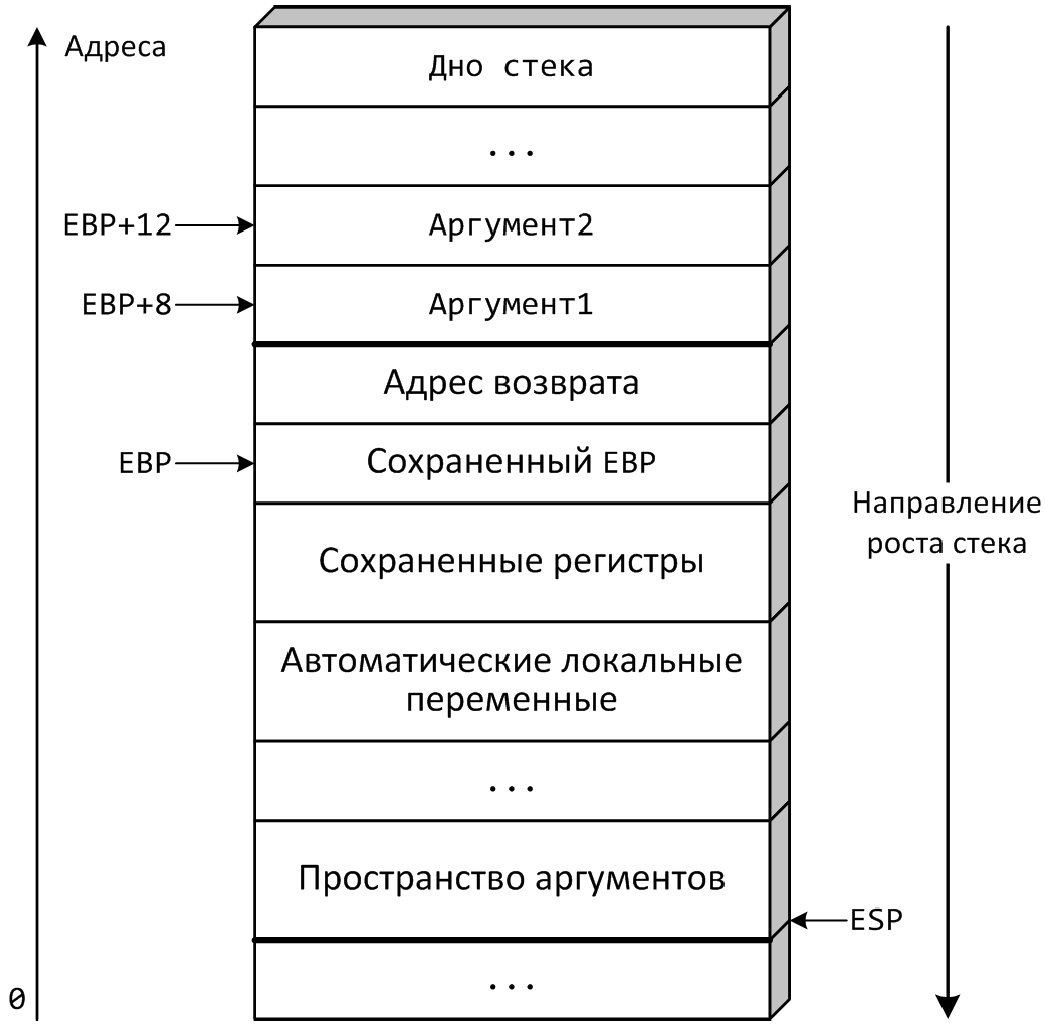
Соглашение о вызове cdecl

- Функция должна сохранять значение регистров `ebx, esi, edi, ebp, esp` - какие значения были перед вызовом, такие должны быть и после
- При необходимости использования регистры сохраняются в стеке



Соглашение о вызове cdecl

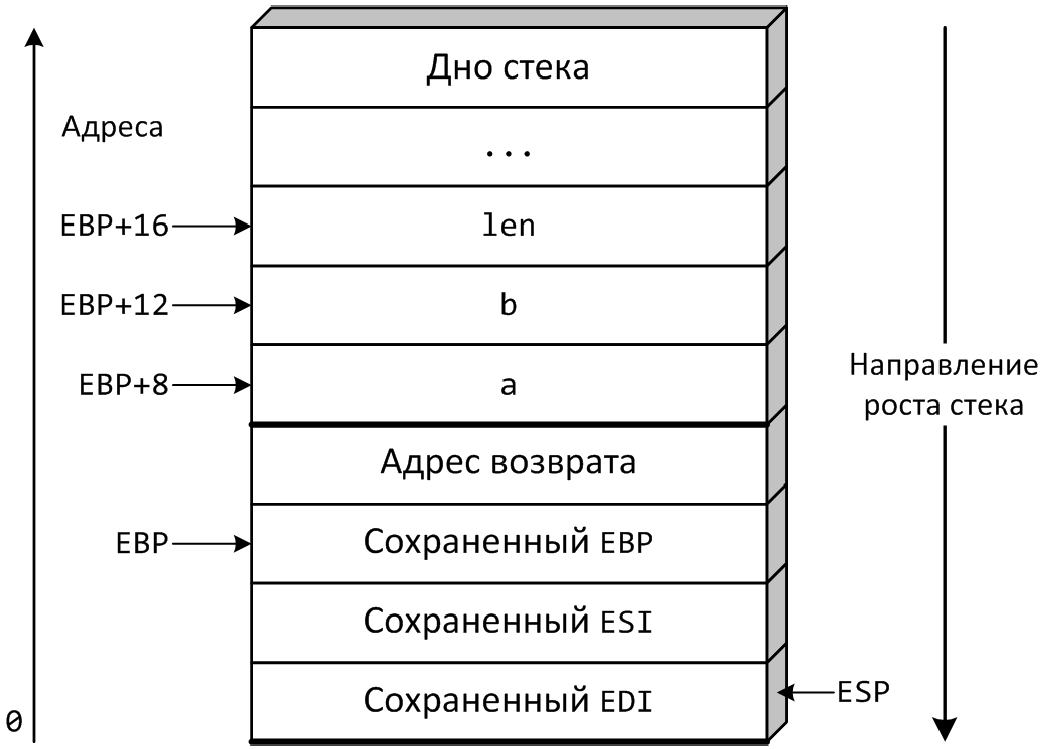
- Если мы вызываем функцию, она может испортить значения регистров eax, ecx, edx
- Если нам нужны эти значения, перед вызовом их следует сохранить в стеке



Пример

Используя соглашение cdecl, реализуйте функцию `conv`, вычисляющую скалярное произведение двух массивов длины `len`.

```
int conv(int *a, int *b, int len);
```



Пример

```
conv:
   push ebp
   mov  ebp, esp            ; Стандартный пролог
   push esi                 ; Сохраняем esi
   push edi                 ; Сохраняем edi
   mov  edi, dword [ebp+8]  ; Извлекаем со стека первый параметр a
   mov  esi, dword [ebp+12] ; Извлекаем со стека второй параметр b
   mov  ecx, dword [ebp+16] ; Извлекаем со стека третий параметр len

   mov  eax, 0              ; В eax будем накапливать скалярное произведение
 .l:
   mov  edx, dword [edi+4*ecx-4]
   imul edx, dword [esi+4*ecx-4]
   add  eax, edx
   loop .l

   pop  edi                 ; Восстанавливаем сохраненный регистр edi
   pop  esi                 ; Восстанавливаем сохраненный регистр esi
   mov  esp, ebp            ; Стандартный эпилог
   pop  ebp
   ret
```

Задача

Требуется реализовать функцию max, соответствующую прототипу `int max(int a, int b)`, которая возвращает максимальное значение своих параметров.

Задача

Требуется оформить функцию `swap`, соответствующую прототипу `void swap(int *a, int *b)`, которая меняет значения своих аргументов.

Пример

Тело Си-функции `f`, использующей соглашение вызова cdecl, состоит из следующего кода.

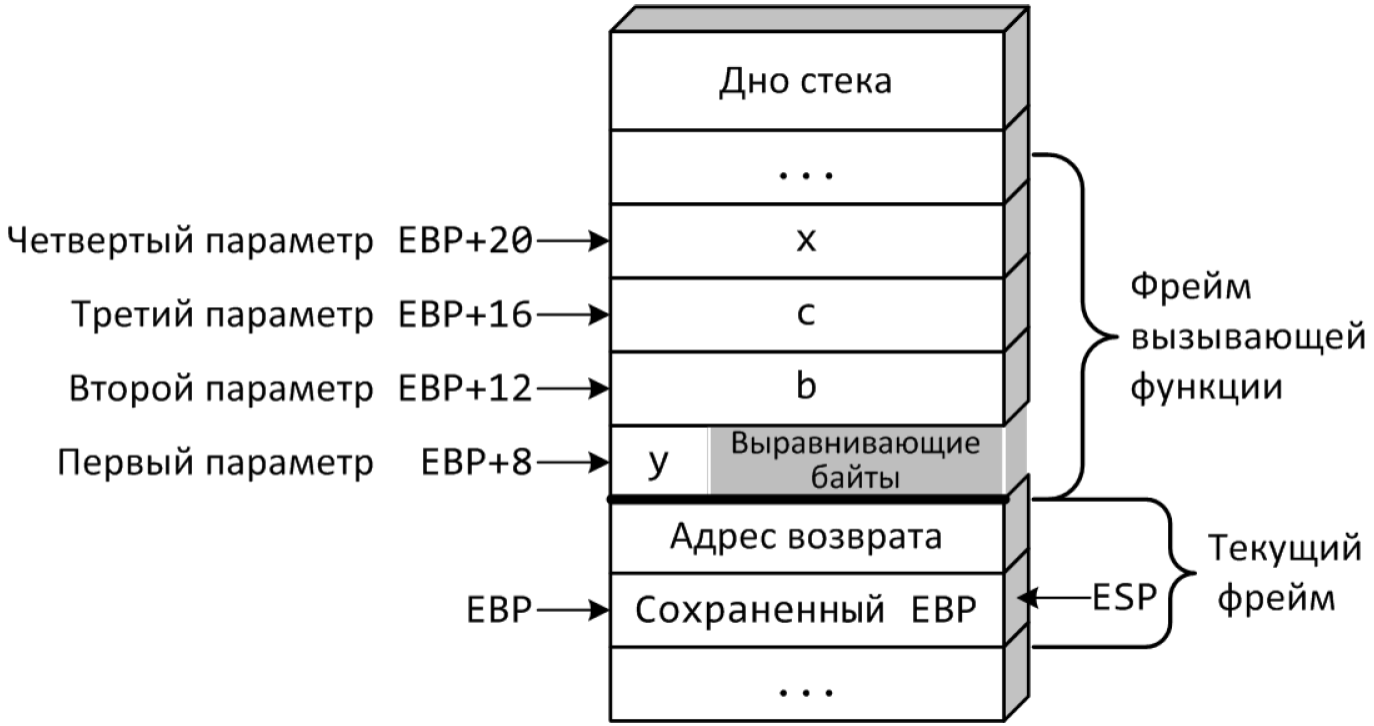
```
*b = x;
return c - y;
```

Компилятор сгенерировал для данного фрагмента следующий ассемблерный код.

```
movsx ecx, byte [ebp+8]    ; (1)
mov   eax, dword [ebp+16]  ; (2)
mov   ebx, dword [ebp+20]  ; (3)
mov   edx, dword [ebp+12]  ; (4)
sub   eax, ecx             ; (5)
mov   word [edx], bx       ; (6)
```

Восстановите прототип функции `f`.

Пример




```
int f(signed char y, short *b, int c, int x);
```

Задача

Функция `h` использует соглашение cdecl. Найти ошибки в реализации, если они есть, и объяснить их, указав, какие именно правила были нарушены.

```
int h(int a, int b) {
   return a % b;
}
```

```
h:
   push ebp
   mov  ebp, esp
   mov  edx, dword [ebp+8]
   mov  eax, edx
   sar  edx, 31
   mov  ebx, dword [ebp+12]
   idiv ebx
   pop  ebp
   ret
```

Пример

В статической переменной `p` хранится указатель на функцию, принимающей в качестве параметров два числа типа `short` и возвращающей `int`. Реализуйте вызов функции по указателю `p` на языке ассемблера, включая объявление переменной `p`. Фактическими аргументами вызова являются числа `0x16` и `0x12`. Для вызова функции используйте соглашение cdecl.

```
section .bss                ; Выделяем 4 байта под указатель
   p resd 1

section .text
   sub  esp, 8              ; Выделяем на стеке место для двух аргументов
   mov  [esp], word 0x16    ; Кладем на стек первый аргумент
   mov  [esp+4], word 0x12  ; Кладем на стек второй аргумент
   call dword [p]           ; Вызываем функцию по указателю p
```

Задача

Реализовать функцию, вычисляющую факториал, через цикл и через рекурсию.