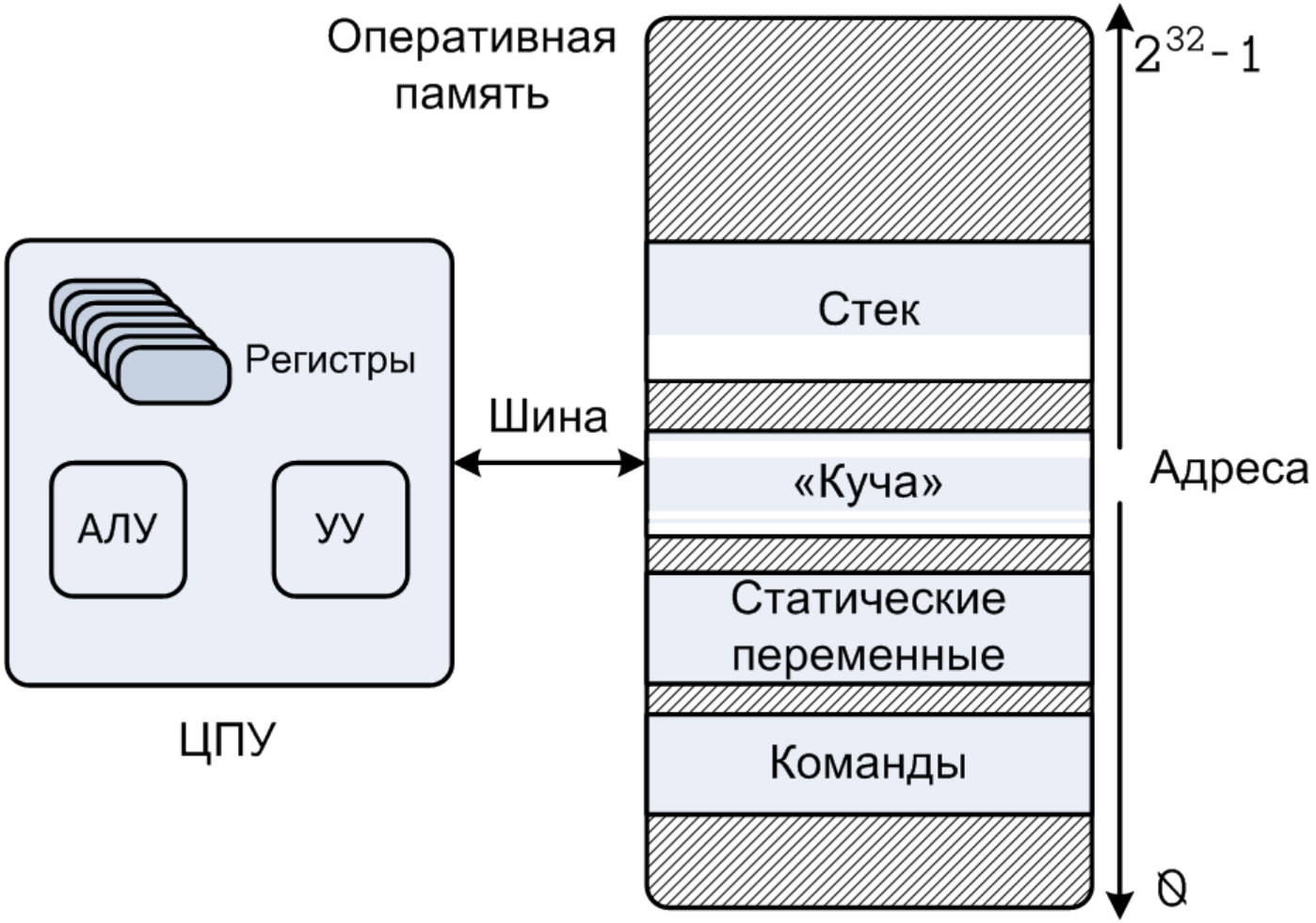
Лекция 2

12 февраля

Машина, на которой работает пользовательская программа (архитектура IA-32)



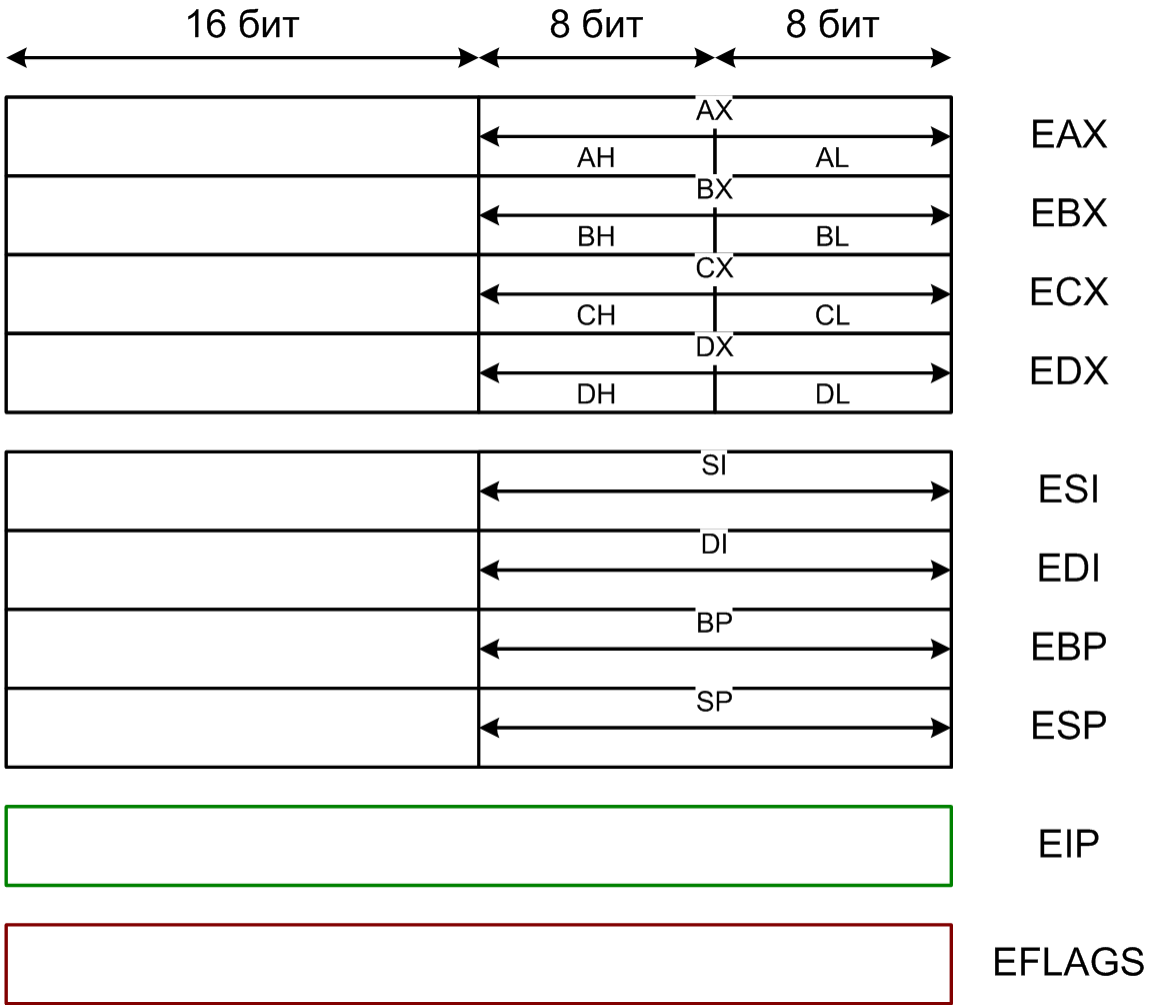
Вспоминаем Си-машину…

Типы операндов

- Регистр r
- Память m
- Константа i

Размеры операндов

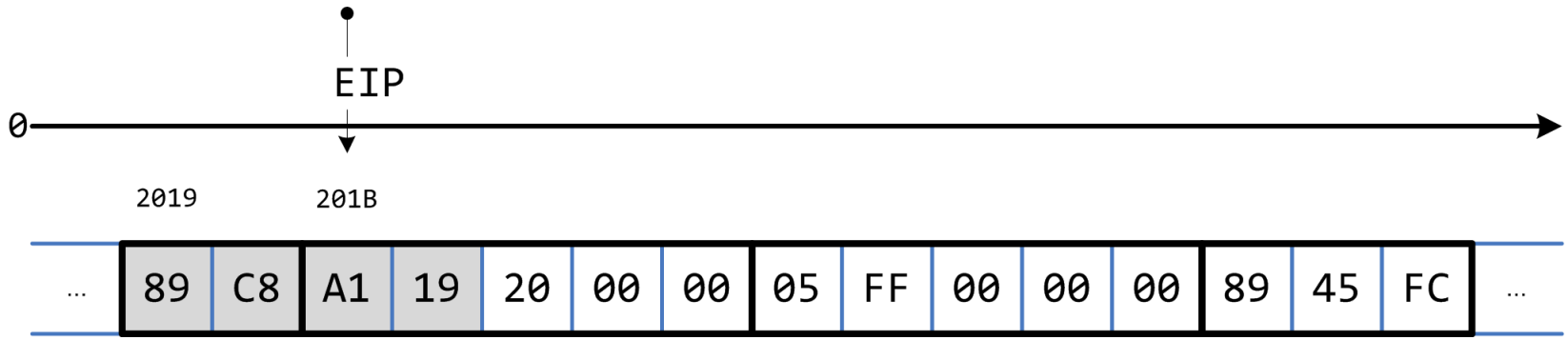
- byte 8
- word 16
- dword 32
- qword 64




```
mov eax, ecx
mov eax, dword [0x2019]
add eax, 0xff
mov dword [ebp-4], eax
```

Где сегментные регистры?

Порядок размещения байт в памяти



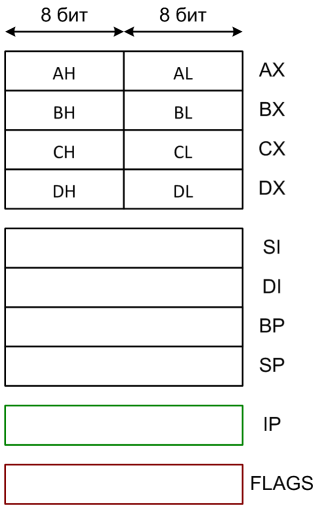

```
mov eax, dword [0x2019]
```

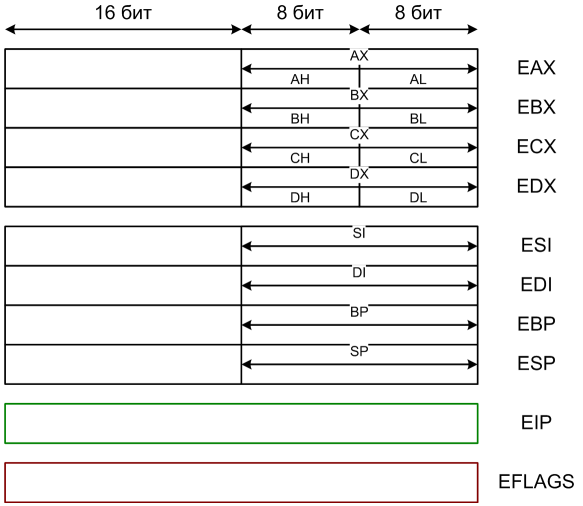
Два наиболее распространенных подхода

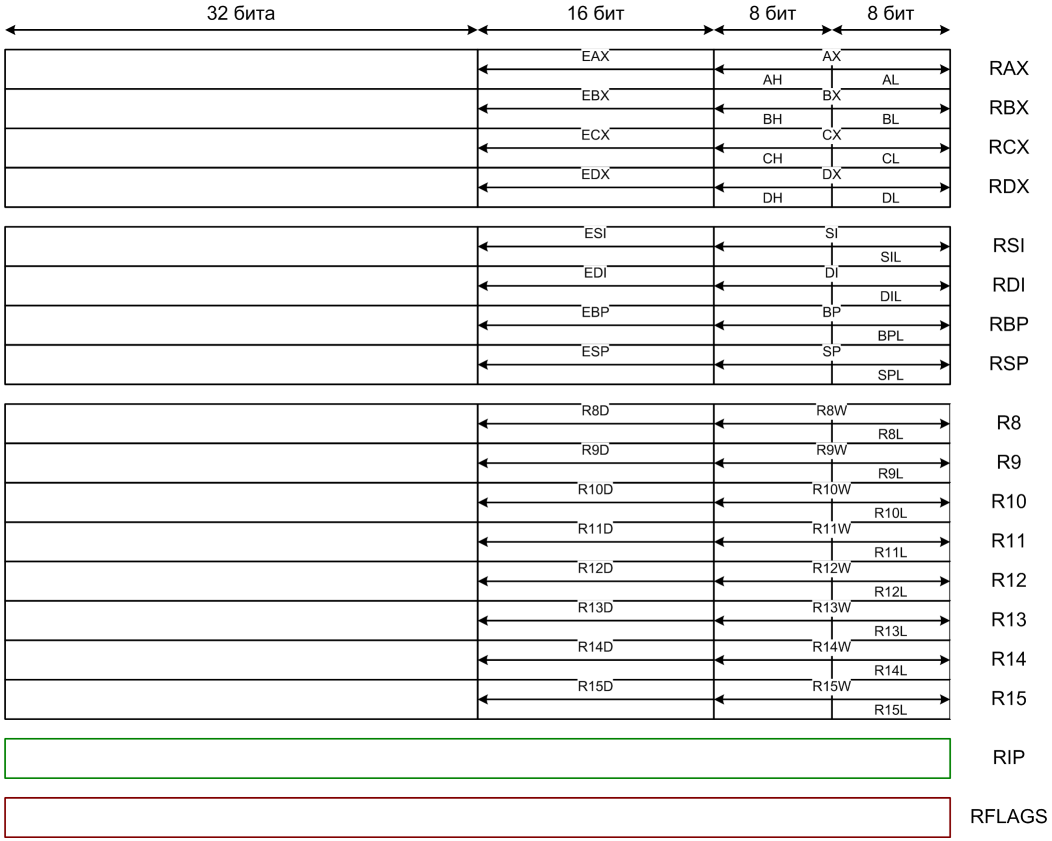
- Порядок от младшего к старшему (*англ. little-endian*) ***используется в IA-32***

- Порядок от старшего к младшему (*англ. big-endian*) как правило используется в процессорах, предназначенных для обработки сетевых данных

8086 → IA-32→ Intel64



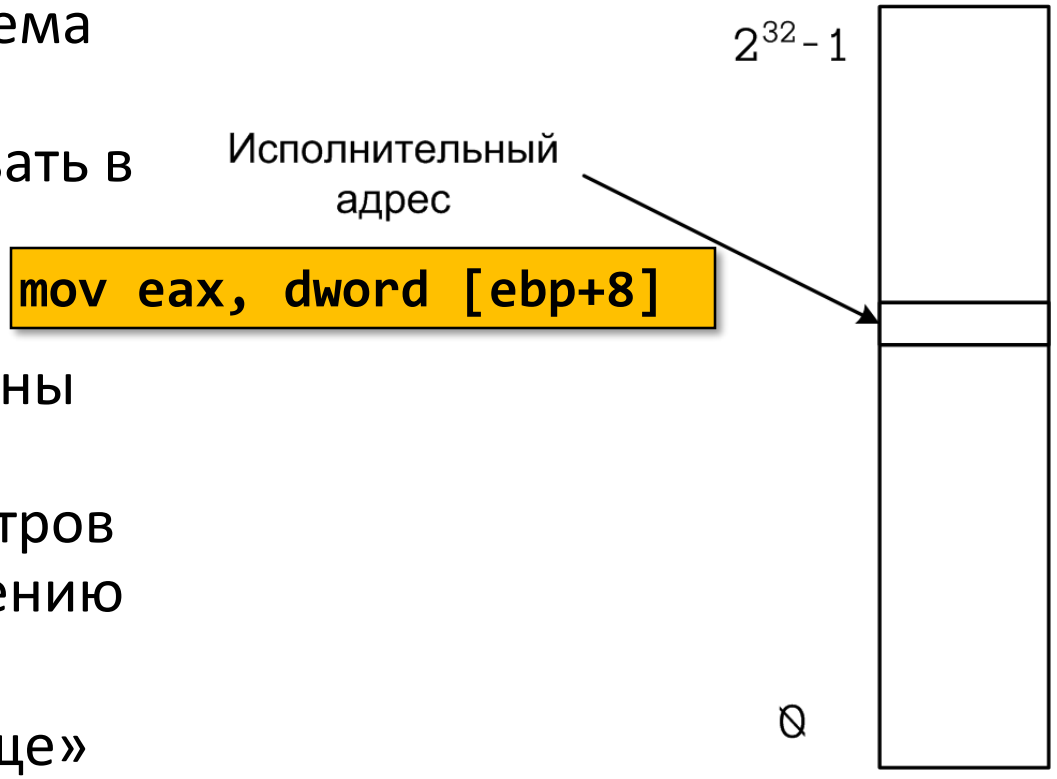




Какие еще названия встречаются?
x86, x86_64, AMD64
IA-64 совсем другая архитектура

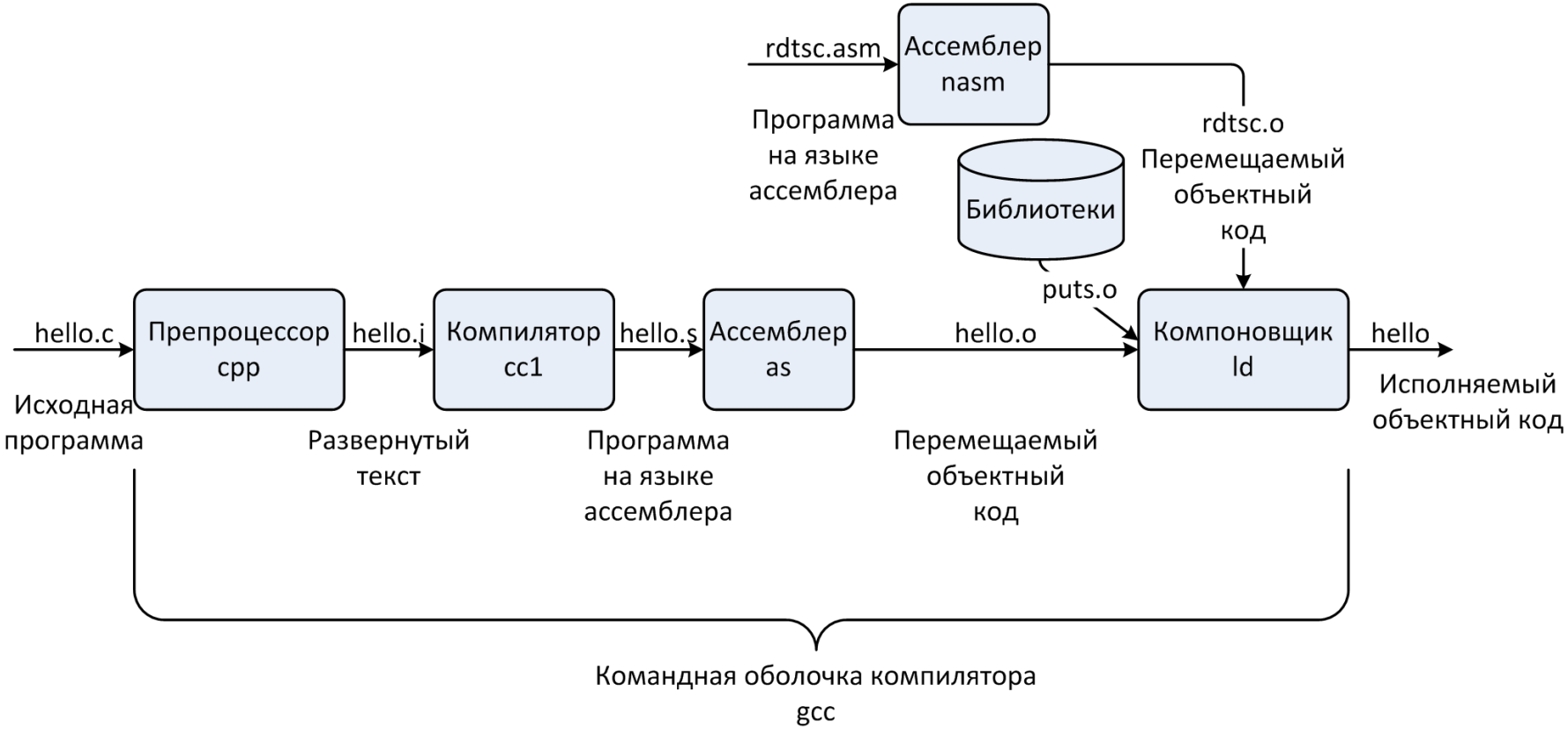
«Положительные» особенности IA-32

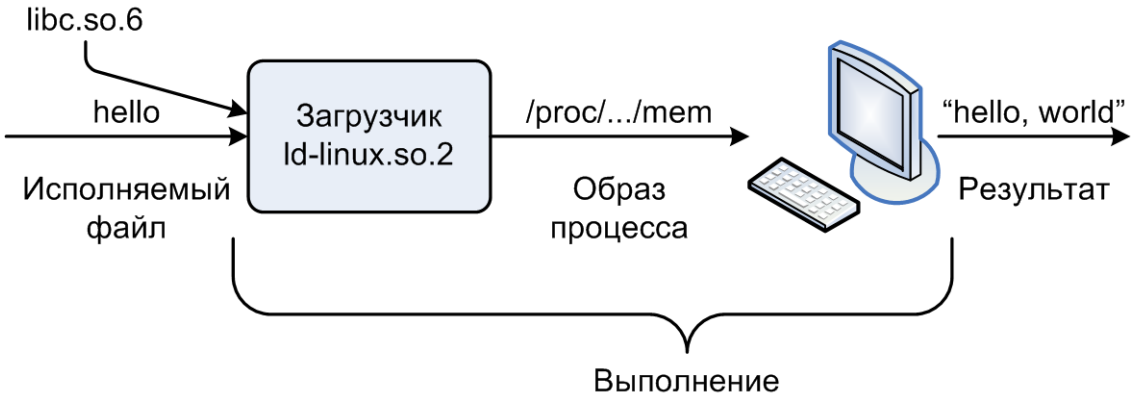
- Особенности аппаратуры и операционная система (Windows/Linux) позволяют использовать в программах модель плоской памяти
- Значительно ослаблены ограничения на использование регистров в командах по сравнению с 8086
- «Количественно проще» чем Intel64



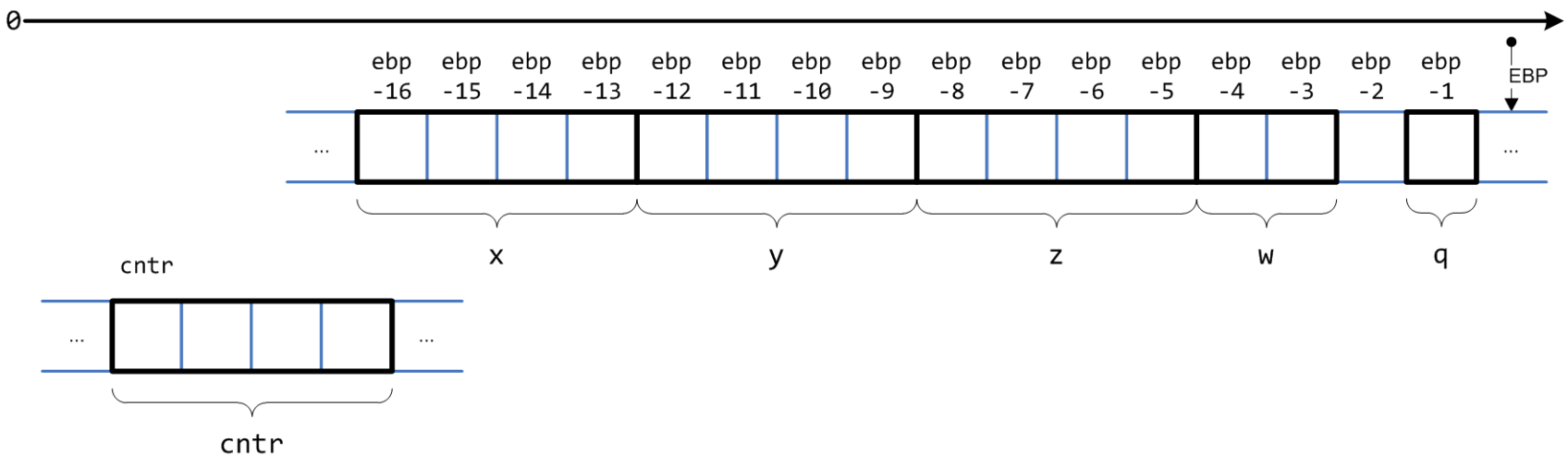
```
void f() {
   static int cntr = 0;      // 1
   int x = 2, y = 1, z = 0; // 2
   unsigned short w = 282;  // 3
   signed char q = 13;      // 4
   ++cntr;                  // 5
   z = -x + q * w *y - w;   // 6
}
```

```
section .bss
; Резервирование 4 байт памяти
   cntr  resd  1
section .text
   global f
; Точка входа в программу
f:
     push    ebp
     mov     ebp, esp
     sub     esp, 16
     mov     dword [ebp-16], 2      ; (1)
     mov     dword [ebp-12], 1      ; (2)
     mov     dword [ebp-8], 0       ; (3)
     mov     word  [ebp-4], 282     ; (4)
     mov     byte  [ebp-1], 13      ; (5)
     add     dword [cntr], 1        ; (6)
     movsx   eax, byte [ebp-1]      ; (7)
     movzx   edx, word [ebp-4]      ; (8)
     imul    eax, edx               ; (9)
     imul    eax, dword [ebp-12]    ; (10)
     sub     eax, dword [ebp-16]    ; (11)
     sub     eax, edx               ; (12)
     mov     dword [ebp-8], eax     ; (13)
     leave
     ret
```



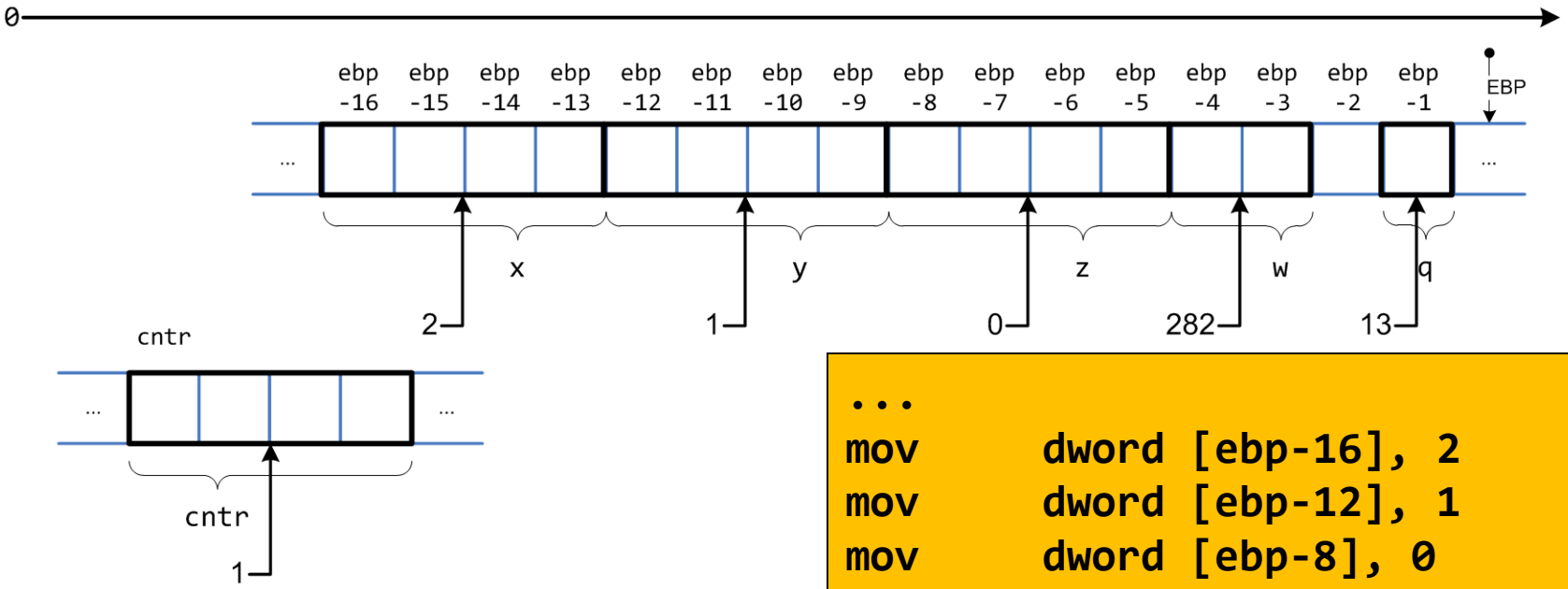


Размещение переменных в памяти



Модели памяти в языке Си

- Автоматическая
  - Каждый вызов функции должен обладать своим комплектом автоматических локальных переменных. Адрес переменной должен определяться динамически, во время работы программы.
- Статическая
  - Один экземпляр переменной. Адрес может быть определен статически, во время компиляции.
- Динамическая
  - Функции стандартной библиотеки языка Си: `malloc` и др.




```
void f() {
   static int cntr = 0;       // 1
   int x = 2, y = 1, z = 0;   // 2
   unsigned short w = 282;    // 3
   signed char q = 13;        // 4
   ++cntr;                    // 5
   z = -x + q * w *y - w;     // 6
}
```

```
...
mov     dword [ebp-16], 2       ; (1)
mov     dword [ebp-12], 1       ; (2)
mov     dword [ebp-8], 0        ; (3)
mov     word  [ebp-4], 282      ; (4)
mov     byte  [ebp-1], 13       ; (5)
add     dword [cntr], 1         ; (6)
movsx   eax, byte [ebp-1]       ; (7)
movzx   edx, word [ebp-4]       ; (8)
imul    eax, edx                ; (9)
imul    eax, dword [ebp-12]     ; (10)
sub     eax, dword [ebp-16]     ; (11)
sub     eax, edx                ; (12)
mov     dword [ebp-8], eax      ; (13)
...
```