Лекция 6

25 февраля

```
; gcc –O2 -S -masm=intel xorcopy.c
...
exchange:
   push    ebp
   mov     ebp, esp
   mov     ecx, dword [ebp+12]
   push    ebx
   mov     ebx, dword [ebp+8]
   mov     eax, dword [ecx]
   mov     edx, eax
   xor     edx, dword [ebx]
   xor     eax, edx
   mov     dword [ecx], eax
   xor     eax, edx
   mov     dword [ebx], eax
   pop     ebx
   pop     ebp
   ret
...
```

```
-bash-2.05b$ gcc -O2 -o xorcopy xorcopy.c
-bash-2.05b$ ./xorcopy
Before x = 0, y = 1
After  x = 1, y = 0
-bash-2.05b$ gcc -O0 -o xorcopy xorcopy.c
-bash-2.05b$ ./xorcopy
Before x = 0, y = 1
After  x = 0, y = 0
```

```
#include <stdio.h>

void exchange(int *x, int *y) {
   *x ^= (*y ^= (*x ^= *y));
}

int main() {
   int x = 0, y = 1;
   printf("Before x = %d, y = %d\n", x, y);
   exchange(&x, &y);
   printf("After  x = %d, y = %d\n", x, y);
   return 0;
}
```

```
; gcc -O0 -S -masm=intel xorcopy.c
...
exchange:
   push    ebp
   mov     ebp, esp
   push    ebx
   mov     eax, dword [ebp+8]
   mov     ebx, dword [eax]
   mov     eax, dword [ebp+12]
   mov     ecx, dword [eax]
   mov     eax, dword [ebp+8]
   mov     edx, dword [eax]
   mov     eax, dword [ebp+12]
   mov     eax, dword [eax]
   xor     edx, eax
   mov     eax, dword [ebp+8]
   mov     dword [eax], edx
   mov     eax, dword [ebp+8]
   mov     eax, dword [eax]
   mov     edx, ecx
   xor     edx, eax
   mov     eax, dword [ebp+12]
   mov     dword [eax], edx
   mov     eax, dword [ebp+12]
   mov     eax, dword [eax]
   mov     edx, ebx
   xor     edx, eax
; в xor используется *x (edx = 0),
; вычисленный до первого присваивания
   mov     eax, dword [ebp+8]
   mov     dword [eax], edx
   pop     ebx
   pop     ebp
   ret
...
```

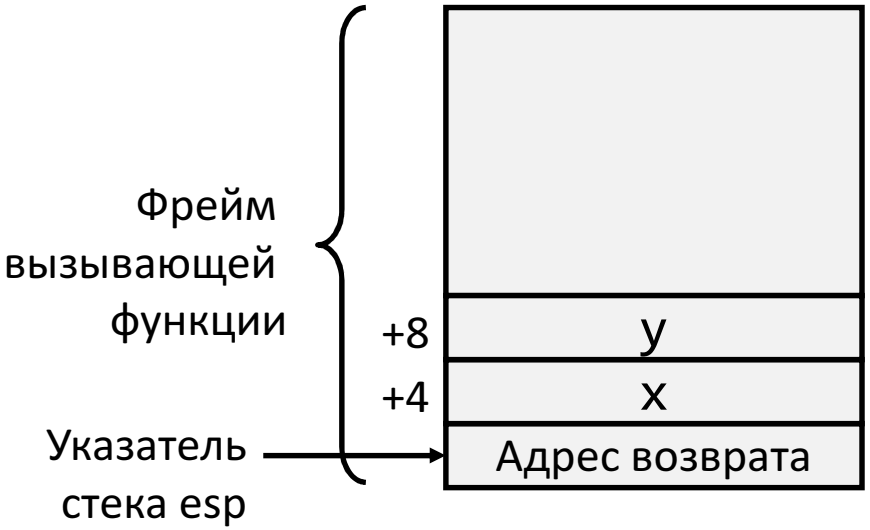
```
-bash-2.05b$ gcc -O2 -o xorcopy xorcopy.c
-bash-2.05b$ ./xorcopy
Before x = 0, y = 1
After  x = 1, y = 0
-bash-2.05b$ gcc -O0 -o xorcopy xorcopy.c
-bash-2.05b$ ./xorcopy
Before x = 0, y = 1
After  x = 0, y = 0
```

```
#include <stdio.h>

void exchange(int *x, int *y) {
   *x ^= (*y ^= (*x ^= *y));
}

int main() {
   int x = 0, y = 1;
   printf("Before x = %d, y = %d\n", x, y);
   exchange(&x, &y);
   printf("After  x = %d, y = %d\n", x, y);
   return 0;
}
```

Пример: обмен значениями с использованием указателей




```
void exchange(int *x, int *y) {
   int tmp = *x;
   *x = *y;
   *y = tmp;
}
```

Регистр	Значение
edx	**tmp**

Фрейм намеренно не создается

```
exchange:
   mov eax, [esp+4] ; eax ← x
   mov edx, [eax]   ; edx ← *x
   mov ecx, [esp+8] ; ecx ← y
   mov ecx, [ecx]   ; ecx ← *y
   mov [eax], ecx   ; *x  ← ecx
   mov eax, [esp+8] ; eax ← y
   mov [eax], edx   ; *y  ← edx
```

```
int tmp = *x;
```

```
*x = *y;
```

```
*y = tmp;
```

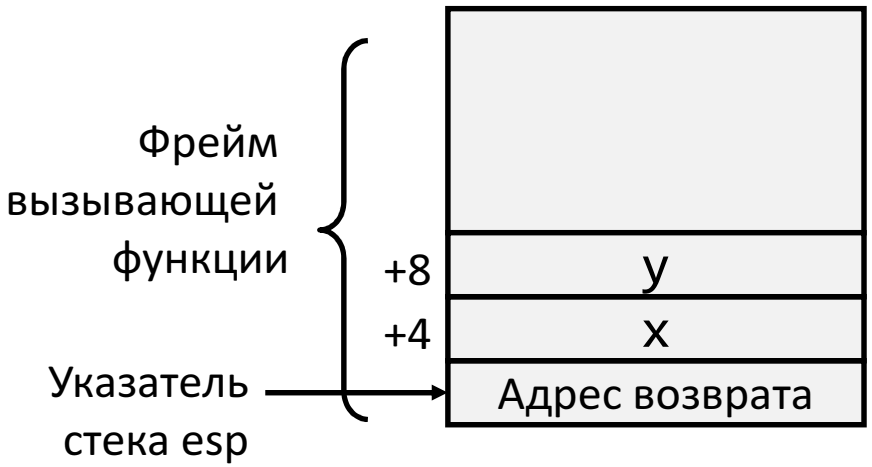
Пример: обмен значениями с использованием указателей

```
void exchange(int *x, int *y) {
   if (x == y) {
      return;
   }

   *x ^= *y;
   *y ^= *x;
   *x ^= *y;
}
```

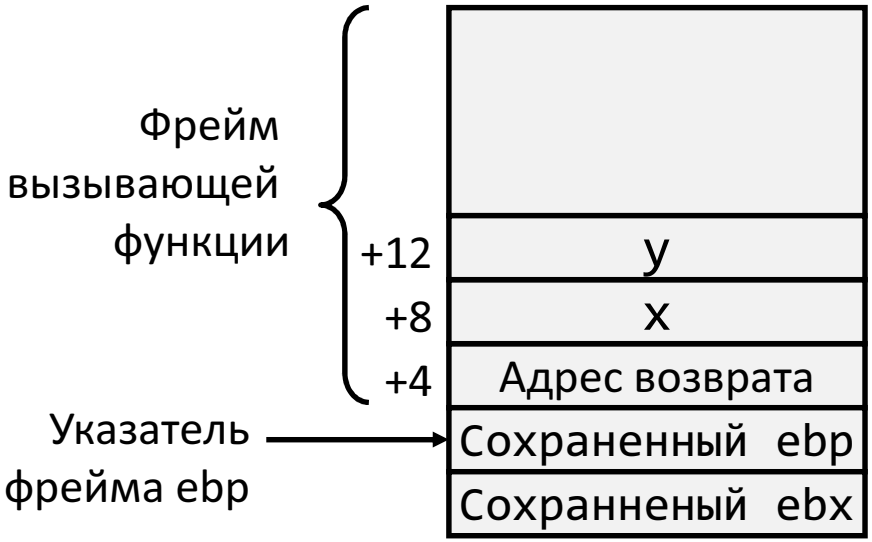
Фрейм намеренно не создается

```
exchange:
   mov     ecx, dword [esp+4]
   mov     edx, dword [esp+8]
   cmp     ecx, edx
   je      .L3
   mov     eax, dword [edx]
   xor     eax, dword [ecx]
   mov     dword [ecx], eax
   xor     eax, dword [edx]
   mov     dword [edx], eax
   xor     dword [ecx], eax
.L3:
   ret
```



Пример: обмен значениями с использованием указателей

```
void exchange(int *x, int *y) {
   int tmp = *x;
   *x = *y;
   *y = tmp;
}
```




```
exchange:
   push ebp
   mov  ebp, esp
   push ebx
   mov  edx, dword [ebp+8]
   mov  ecx, dword [ebp+12]
   mov  ebx, dword [edx]
   mov  eax, dword [ecx]
   mov  dword [edx], eax
   mov  dword [ecx], ebx
   pop  ebx
   pop  ebp
   ret
```

Обратная задача

```
void f(int *xp, int *yp, int *zp) {
        ???
}
```

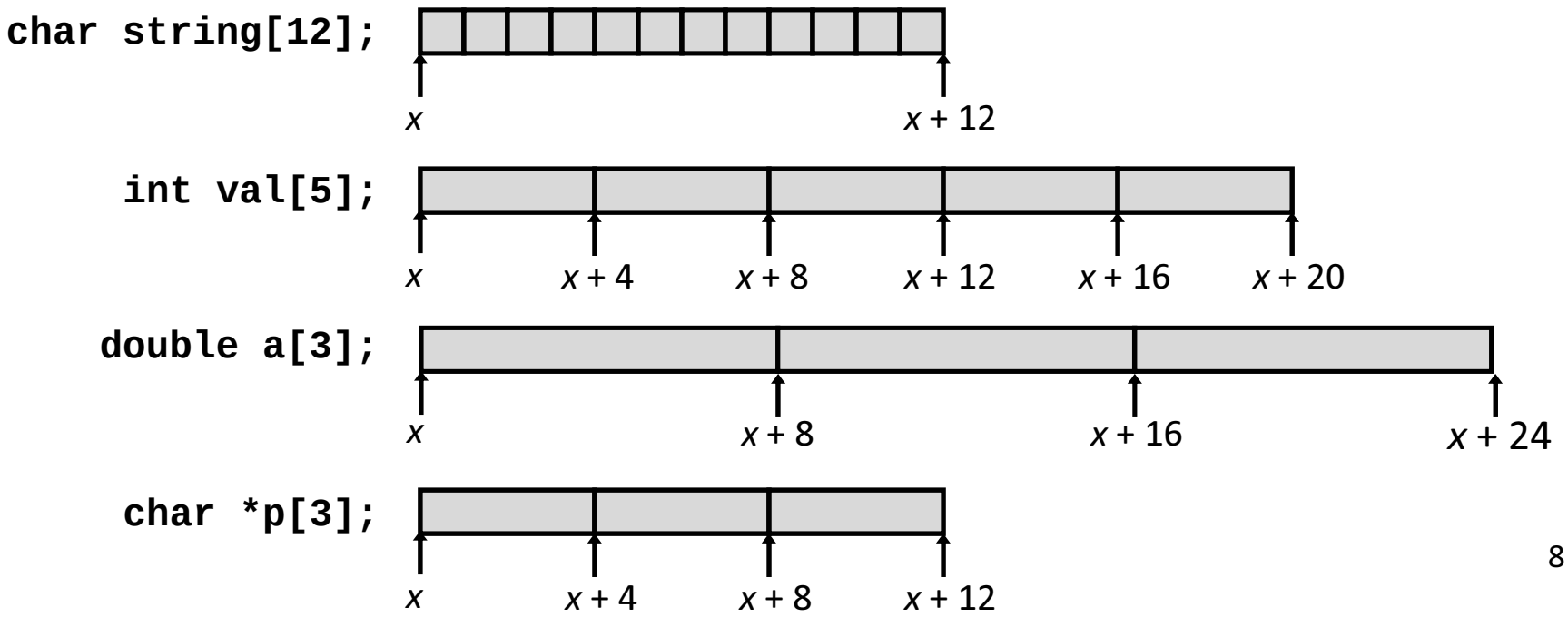
```
exchange:
   ... ; пролог функции
   mov edi, [ebp + 8]   ; (1)
   mov edx, [ebp + 12]  ; (2)
   mov ecx, [ebp + 16]  ; (3)
   mov ebx, [edx]       ; (4)
   mov esi, [ecx]       ; (5)
   mov eax, [edi]       ; (6)
   mov [edx], eax       ; (7)
   mov [ecx], ebx       ; (8)
   mov [edi], esi       ; (9)
   ... ; эпилог функции
```

Параметр	Размещение
xp	**ebp + 8**
yp	**ebp + 12**
zp	**ebp + 16**

Массивы – размещение в памяти

`T A[L];`

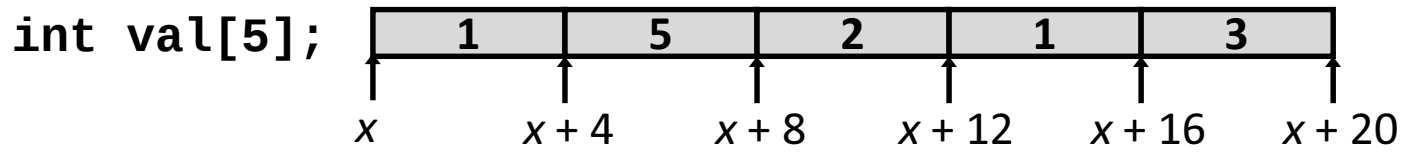
- Массив элементов типа T, размер массива – L
- Массив располагается в непрерывном блоке памяти размером *L* * **`sizeof`**(*T*) байт



Доступ к элементам массива

`T A[L];`

- Массив элементов типа T, размер массива – L
- Идентификатор A может использоваться как указатель на элемент массива с индексом 0. Тип указателя – T*



- Задачи …

Две обратные задачи

```
f:
    push    ebp
    mov     ebp, esp
    mov     edx, dword [ebp+8]
    movsx   ax, byte [a+edx]
    mov     word [b+edx+edx], ax
    pop     ebp
    ret

g:
    push    ebp
    mov     ebp, esp
    mov     edx, dword [ebp+8]
    movzx   eax, word [b+edx+edx]
    mov     byte [a+edx], al
    pop     ebp
    ret
```

```
#define N 256

_______ a[N];
_______ b[N];

void f(______ i) {
   __________;
}

void g(______ i) {
   __________;
}
```