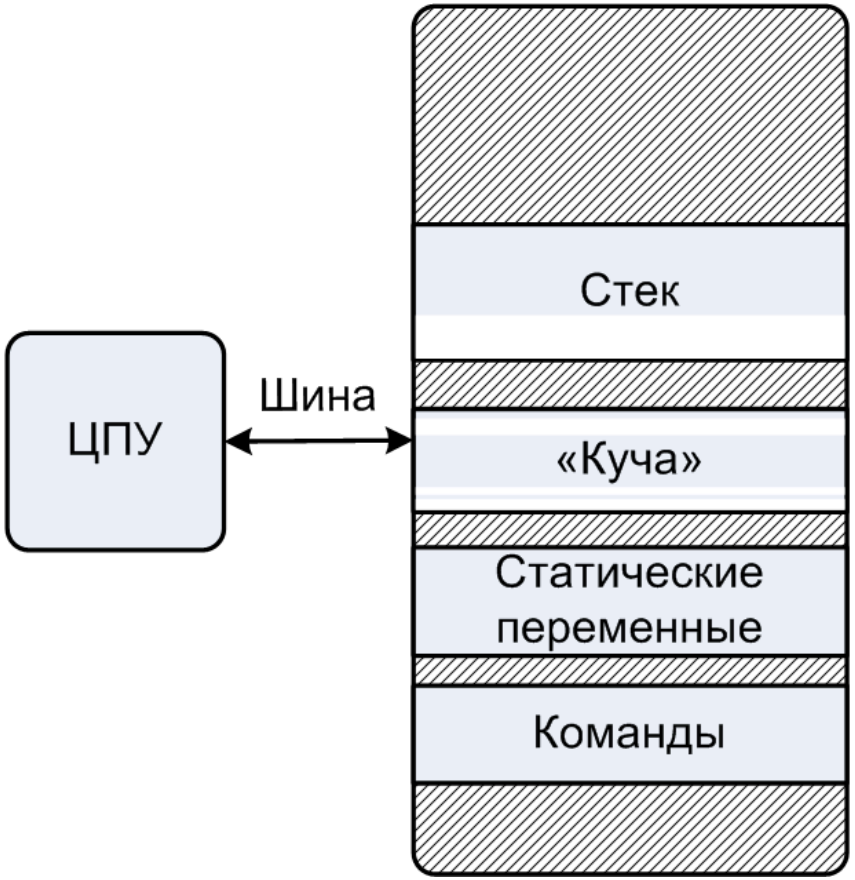
Лекция 8

02 марта

Распределение памяти

- Модели памяти в языке Си
  - Автоматическая
  - Статическая
  - Динамическая
- Секции кода и данных в языке ассемблера
  - .text
  - .data, .bss
  - .rodata



```
-bash-2.05b$ ./build_asm.sh hello.asm
-bash-2.05b$ objdump -h hello
```

```
%include "io.inc"
section .rodata
  msg db `Hello, world!\n`, 0
section .text
  global CMAIN
CMAIN:
  PRINT_STRING msg
  xor eax, eax
  ret
```

```
hello:     file format elf32-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
...
 11 .text         000001cc  08048310  08048310  00000310  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
...
 13 .rodata       00000017  080484f8  080484f8  000004f8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
...
 15 .data         00000008  08049514  08049514  00000514  2**2
                  CONTENTS, ALLOC, LOAD, DATA
...
 21 .bss          0000000c  08049610  08049610  00000610  2**2
                  ALLOC
...
```

Типы данных языка Си

- `char`
- Стандартные знаковые целочисленные типы
  - `signed char`
  - `short int`
  - `int`
  - `long int`
  - `long long int`
- Стандартные беззнаковые целочисленные типы
  - `_Bool`
- Перечисление
- Типы чисел с плавающей точкой
  - `float`
  - `double`
  - `long double`
  - `_Complex`
- Производные типы
  - Массивы
  - Структуры
  - Объединения
  - Указатели
  - Указатели на функции

Регистры и типы данных

- Целые числа
  - Размещаются и обрабатываются в регистрах общего назначения
  - Знаковые/беззнаковые числа

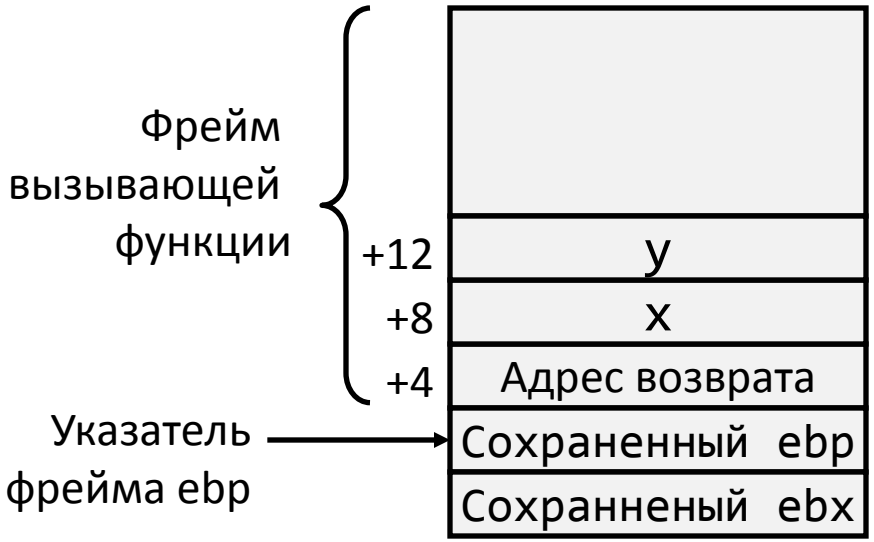
Intel	ASM	Bytes	C
byte	**b**	1	**[unsigned] char**
word	**w**	2	**[unsigned] short**
double word	**d**	4	**[unsigned] int**
quad word	**q**	8	**[unsigned] long long int**

- Указатели
- Числа с плавающей точкой
  - Размещаются и обрабатываются в специализированных регистрах для чисел с плавающей точкой

Intel	ASM	Bytes	C
Single	**d**	4	**float**
Double	**q**	8	**double**

Пример1: 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
```

Пример2: обмен значениями

Используем указатели без дополнительной переменной

```
; 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
...
```

```
#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;
}
```

```
-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
```

Результат изменился!!!

```
; gcc -O0 -S -masm=intel xorcopy.c
...
   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
...
```

```
#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;
}
```

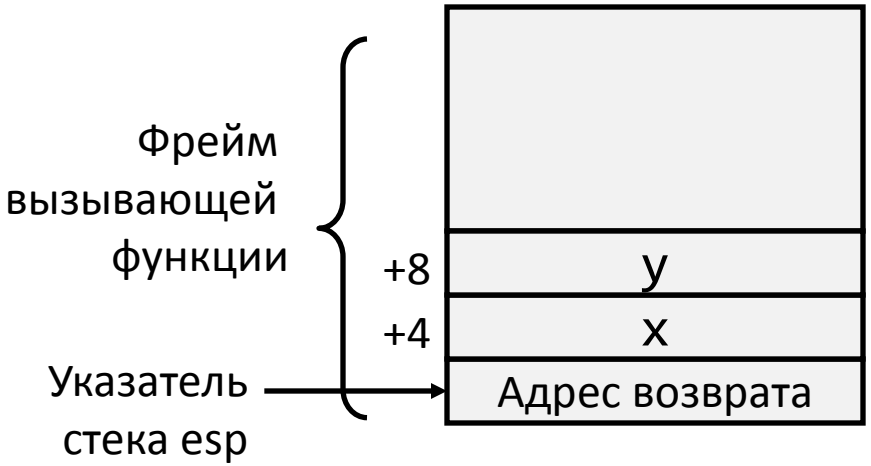
```
-bash-2.05b$ gcc -O2 -o xorcopy xorcopy.c
-bash-2.05b$ ./xorcopy
Before x = 0, y = 1
After  x = 1, y = 0
-bash-2.05b$ gcc -O0 -o xorcopy xorcopy.c
-bash-2.05b$ ./xorcopy
Before x = 0, y = 1
After  x = 0, y = 0
```

Пример3: правильный обмен с использованием команд XOR

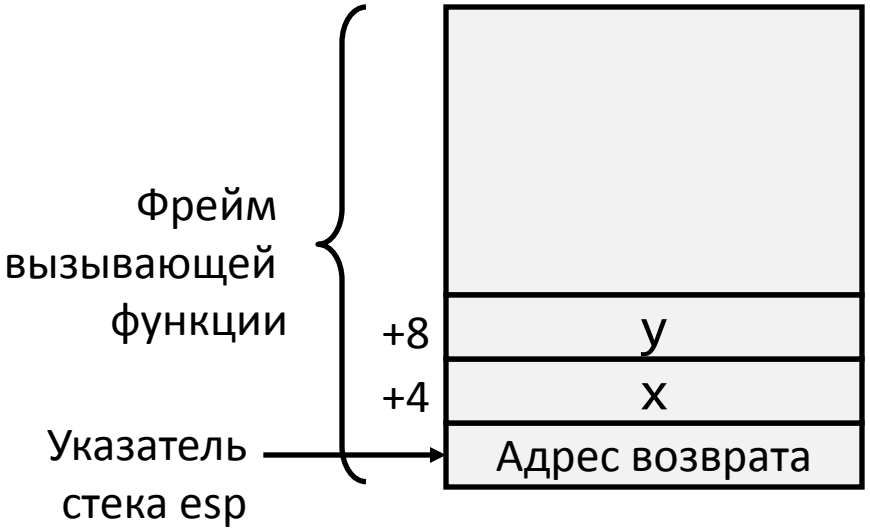
```
void exchange(int *x, int *y) {
   if (x == y) {
      return;
   }

   *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
```



Пример4: 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;
```