© 2013 МГУ/ВМиК/СП

Лекция 5

20 февраля

© 2013 МГУ/ВМиК/СП

Языки программирования (ЯП), базирующиеся на стеке вызовов

- ЯП с поддержкой рекурсии
  - C, Pascal, Java
  - Код функции можно вызывать повторно (“Reentrant”)
    - Одновременно могут выполняться несколько вызовов функции
  - Необходимо выделять память под сохранение состояния каждого работающего вызова
    - Аргументы
    - Локальные переменные
    - Адрес возврата
- Стек
  - Сохранять состояние вызова функции надо в ограниченный период времени: от момента вызова до момент выхода
  - Вызываемая функция всегда завершается до вызывающей
- Стек выделяется **Фреймами**
  - Состояние отдельного вызова функции

© 2013 МГУ/ВМиК/СП

Порядок вызова функции

- Аппаратный стек используется для вызова функций и возврата из них
- Вызов функции: **call label**
  - На стек помещается адрес возврата
  - Выполняется прыжок на метку *label*
- Адрес возврата:
  - Адрес инструкции непосредственно расположенной за инструкцией call

```
804854e:   e8 3d 06 00 00      call  8048b90 <main>
8048553:   50                  push  eax
```

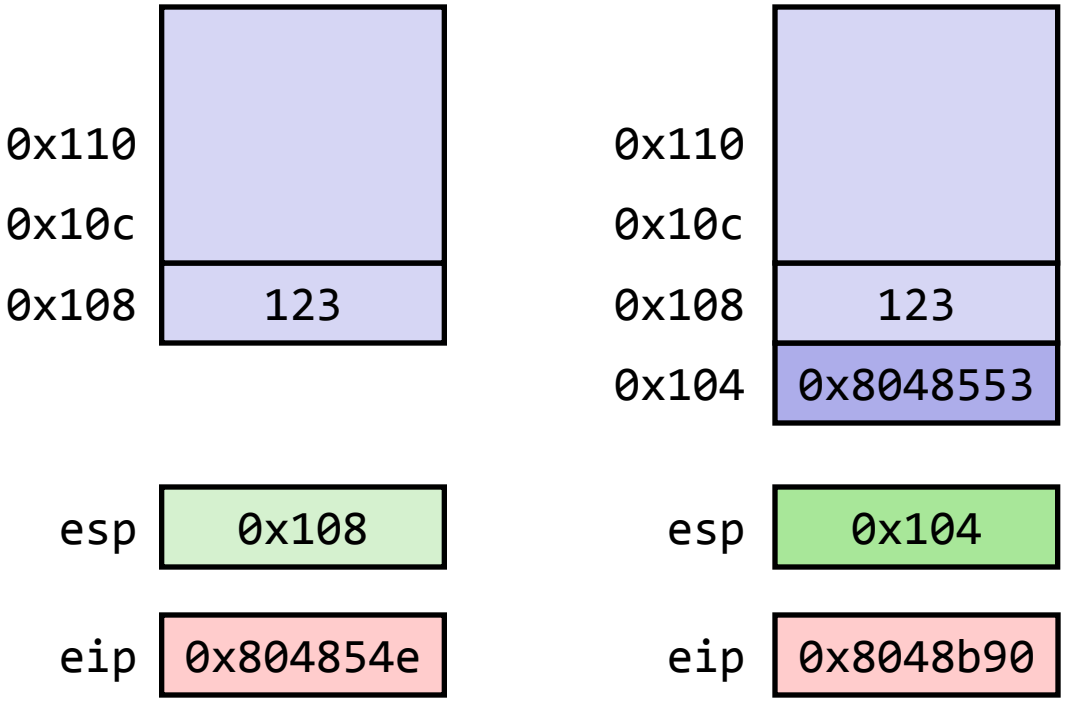
  - Адрес возврата = 0x8048553
- Возврат из функции: **ret**
  - Выгрузка адреса из стека
  - Прыжок на этот адрес

© 2013 МГУ/ВМиК/СП

Вызов функции

```
804854e:        e8 3d 06 00 00          call   8048b90 <main>
8048553:        50                      push   eax
```

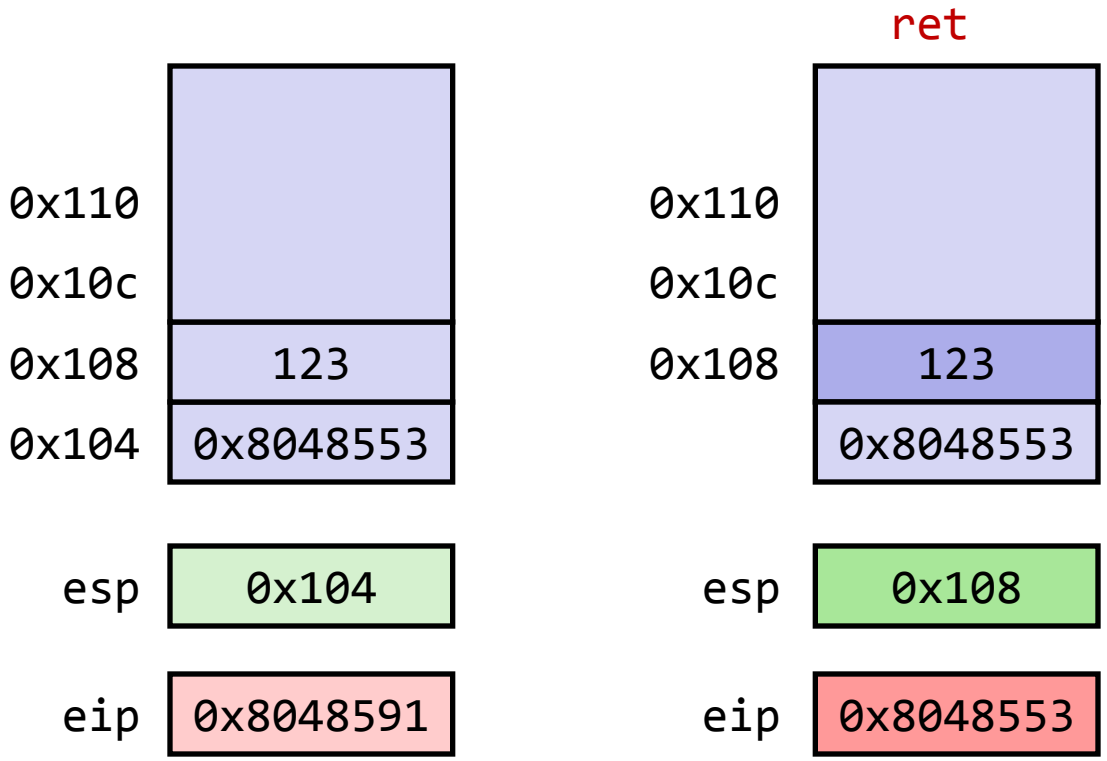
call 8048b90



© 2013 МГУ/ВМиК/СП

Выход из функции

```
8048591:        c3                      ret
```



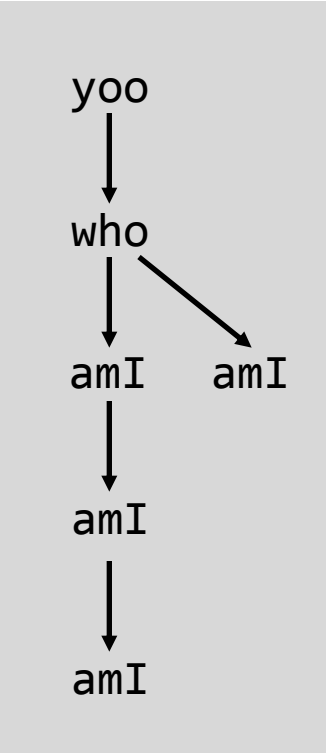
© 2013 МГУ/ВМиК/СП

Пример цепочки вызовов

```
yoo(…)
{
  •
  •
  who();
  •
  •
}
```

```
who(…)
{
  • • •
  amI();
  • • •
  amI();
  • • •
}
```

```
amI(…)
{
  •
  •
  amI();
  •
  •
}
```

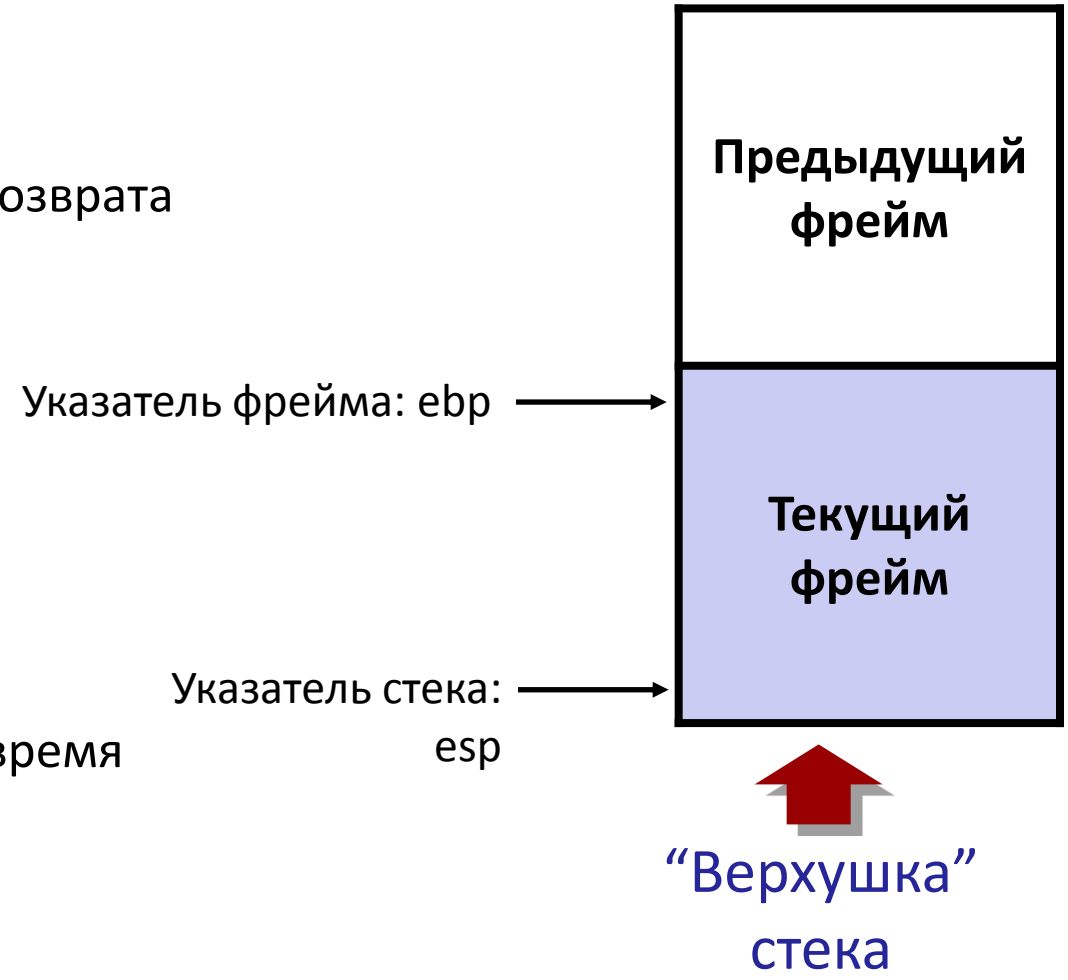


Функция amI() рекурсивная

© 2013 МГУ/ВМиК/СП

Стек фреймов

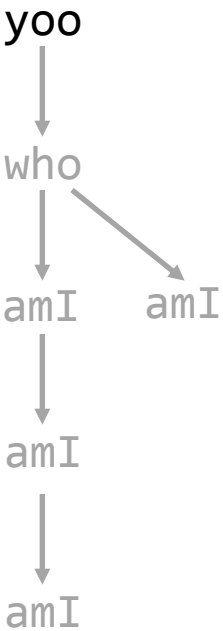
- Во фрейме размещаются
  - Локальные переменные
  - Данные, необходимые для возврата из функции
  - Временные переменные

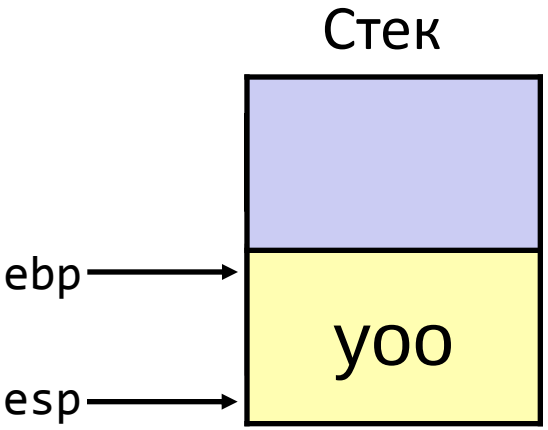


- Управление фреймами
  - Пространство выделятся во время входа в функцию
    - «пролог» функции
  - Освобождается на выходе
    - «эпилог» функции

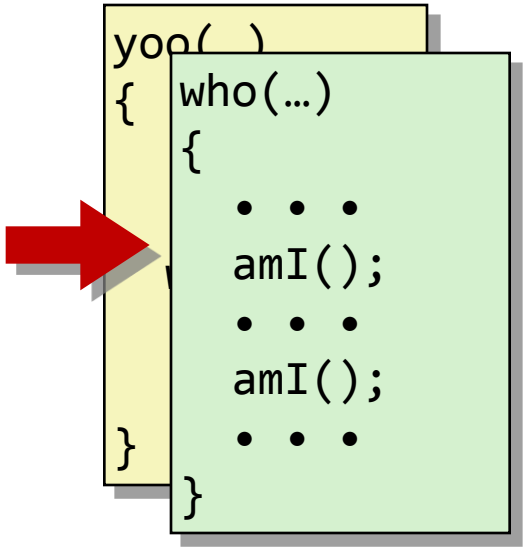
© 2013 МГУ/ВМиК/СП

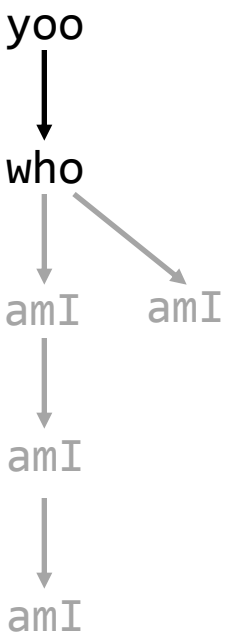
```
yoo(…)
{
   •
   •
   who();
   •
   •
}
```

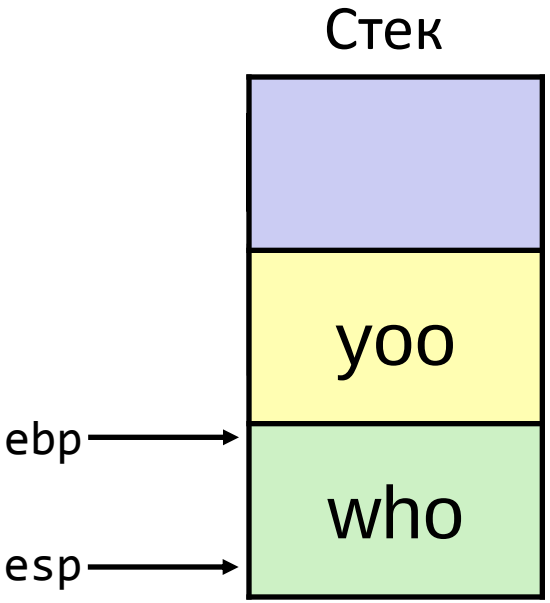




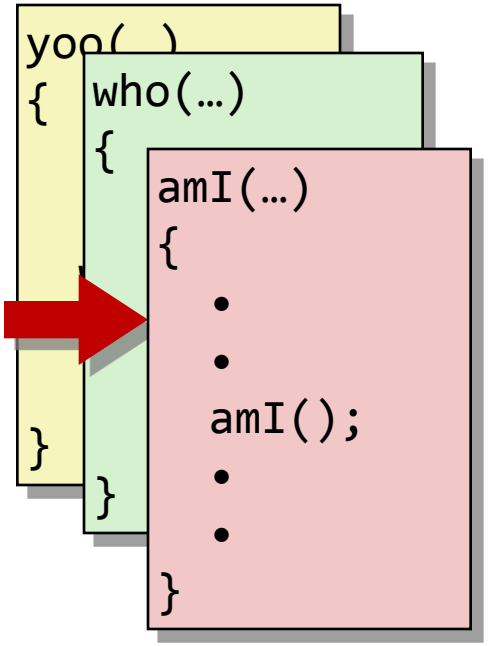
© 2013 МГУ/ВМиК/СП

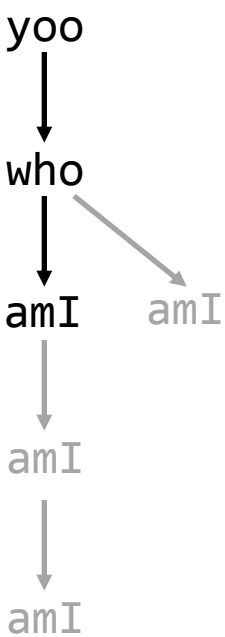


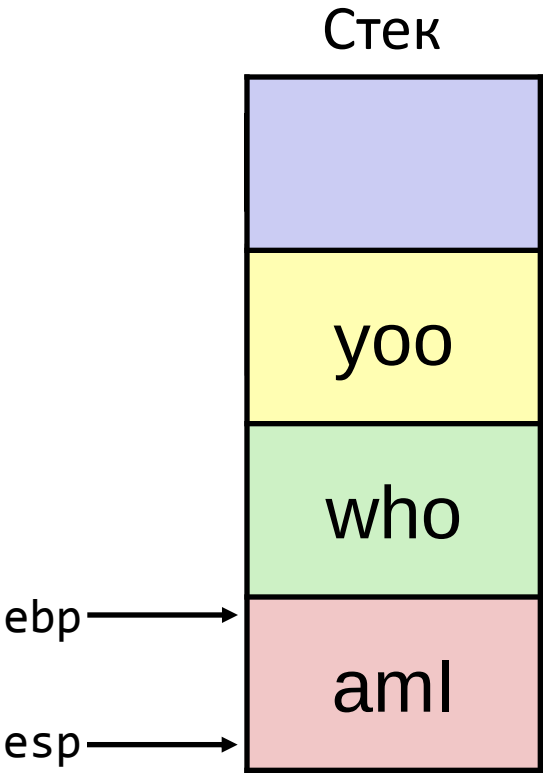




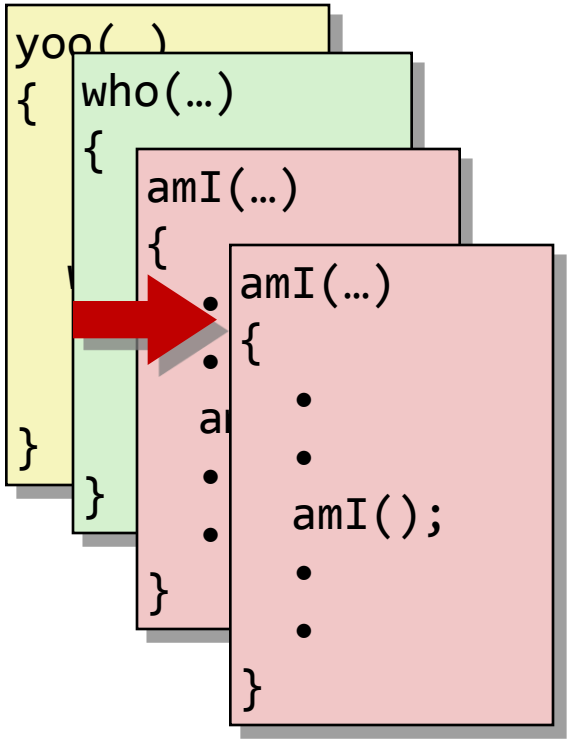
© 2013 МГУ/ВМиК/СП

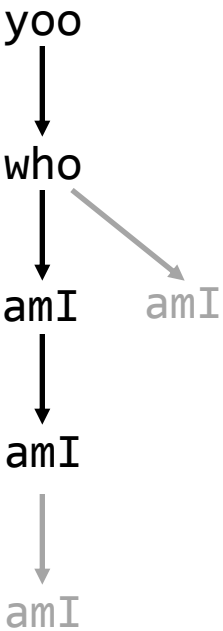


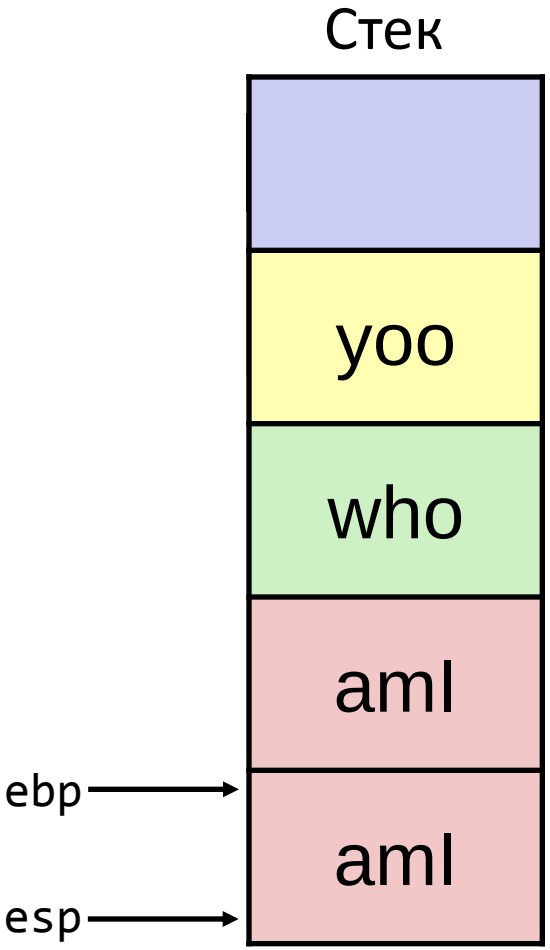




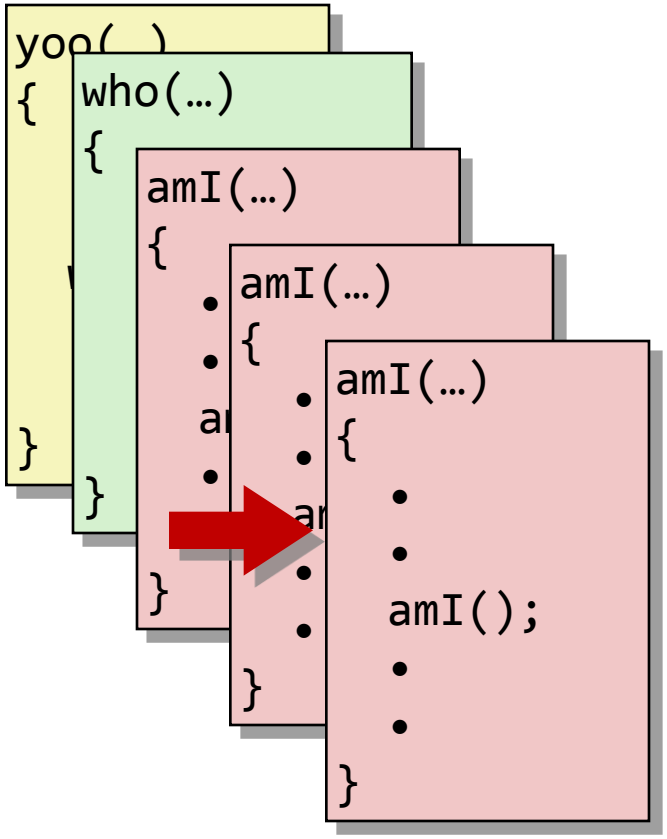
© 2013 МГУ/ВМиК/СП

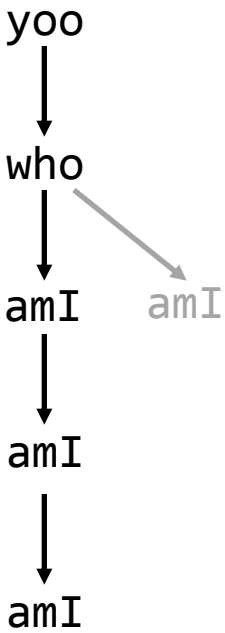




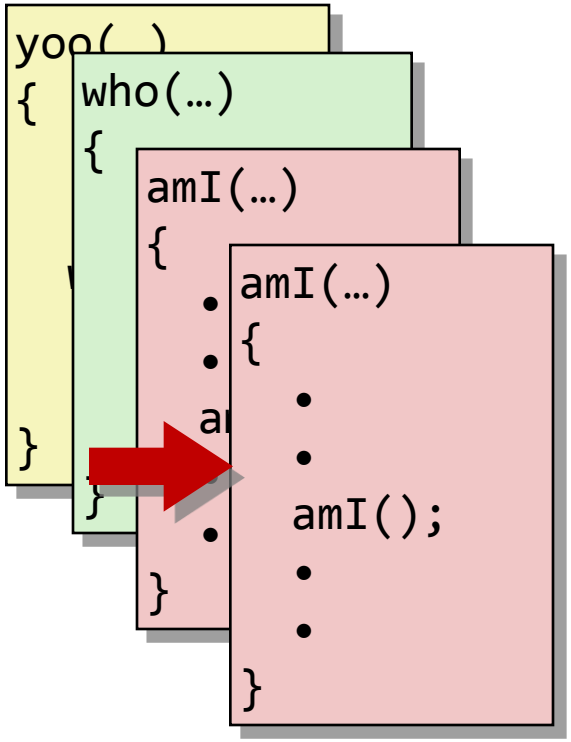


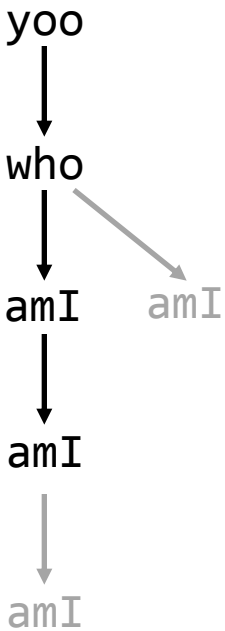
© 2013 МГУ/ВМиК/СП

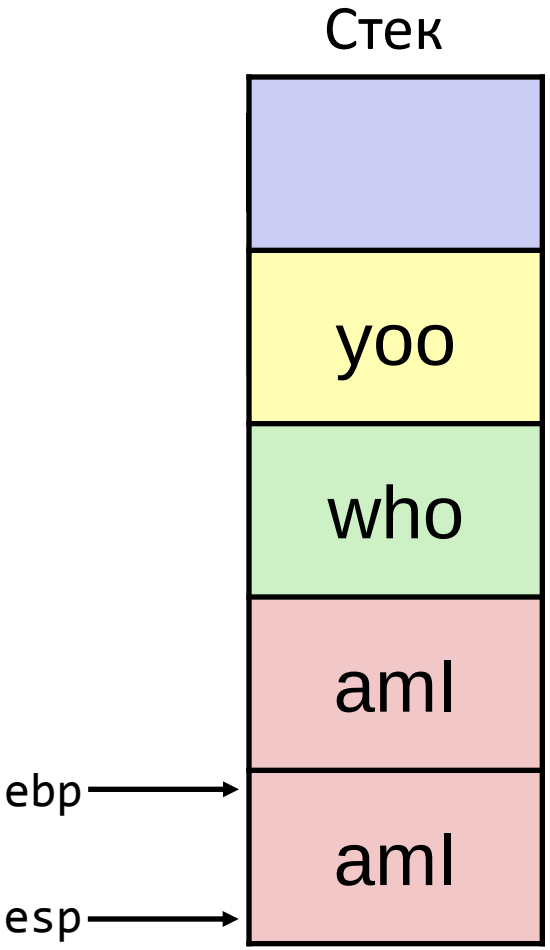




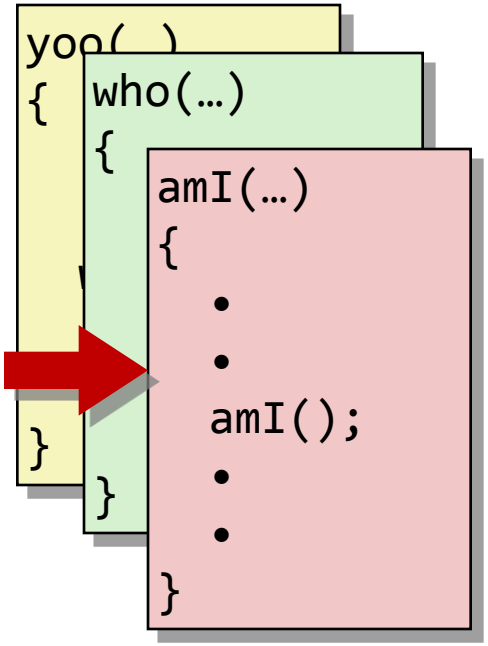
© 2013 МГУ/ВМиК/СП

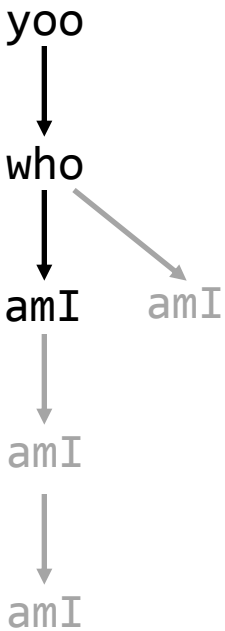


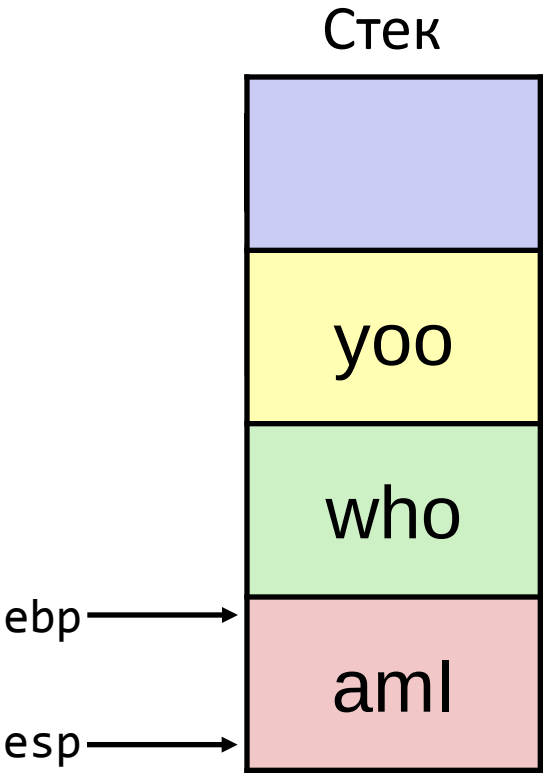




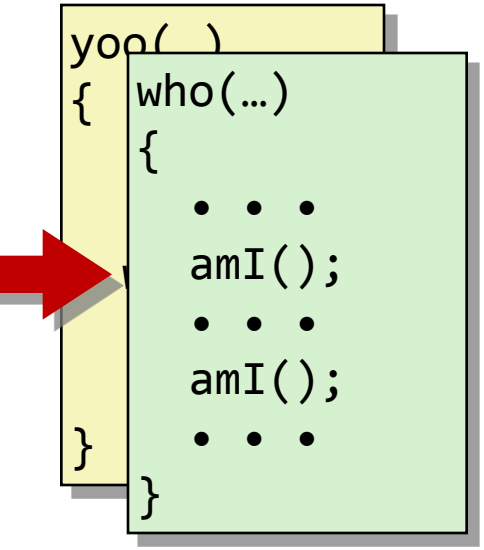
© 2013 МГУ/ВМиК/СП

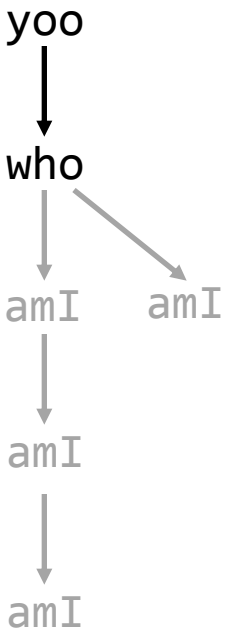


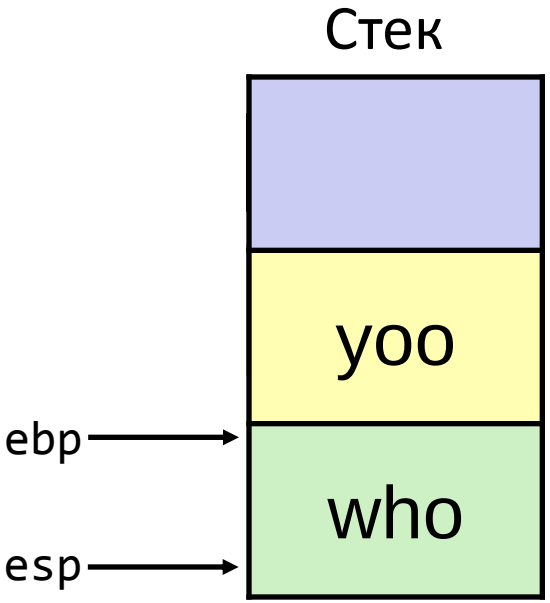




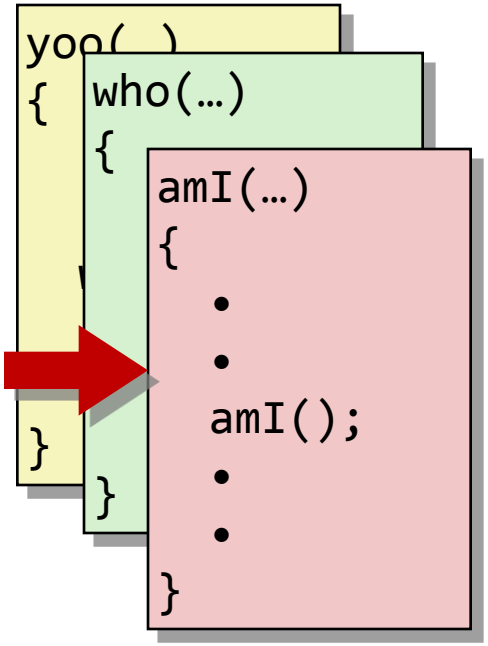
© 2013 МГУ/ВМиК/СП

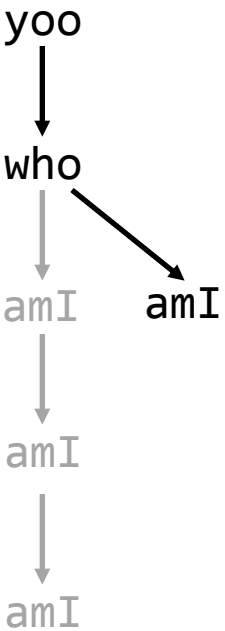


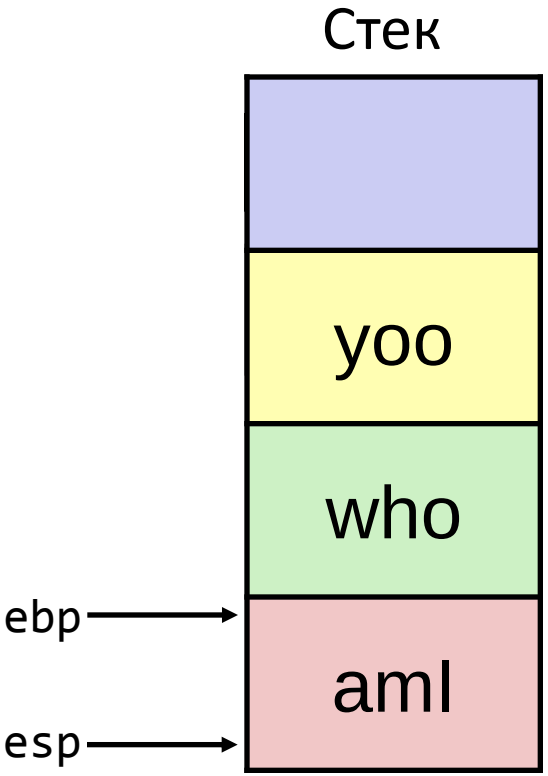




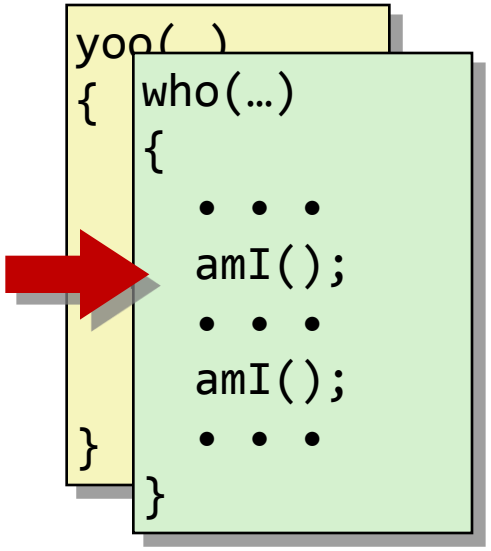
© 2013 МГУ/ВМиК/СП

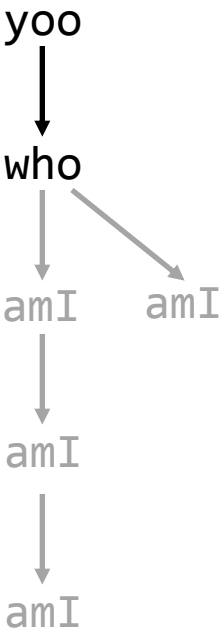


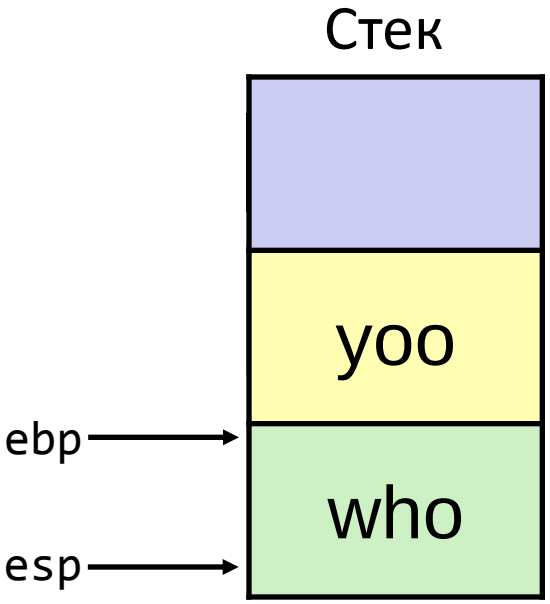




© 2013 МГУ/ВМиК/СП

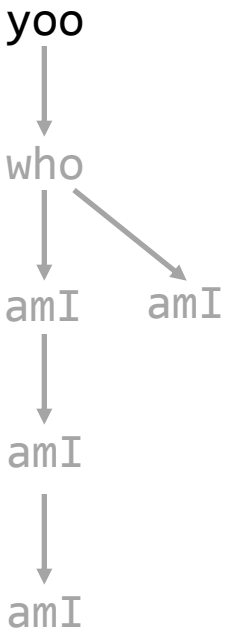


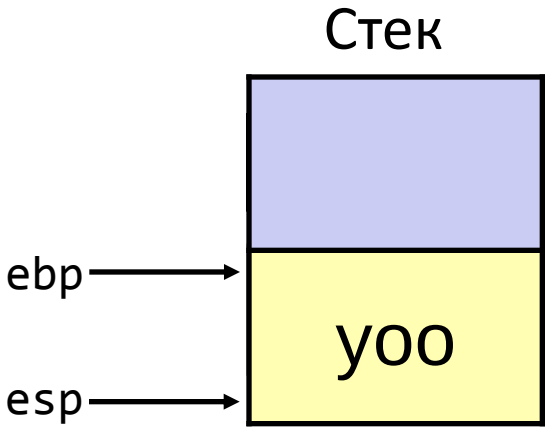




© 2013 МГУ/ВМиК/СП

```
yoo(…)
{
   •
   •
   who();
   •
   •
}
```

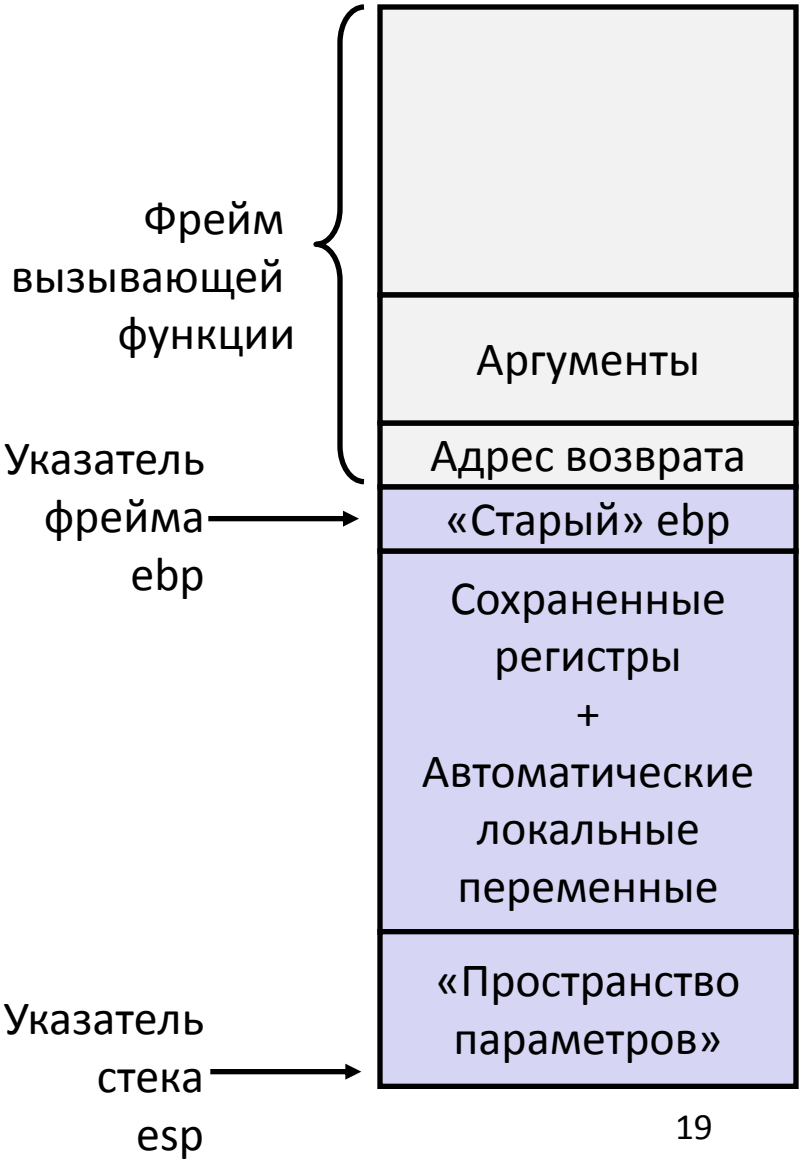




© 2013 МГУ/ВМиК/СП

Организация фрейма в IA32/Linux

- Текущий фрейм (от “верхушки” ко «дну»)
  - “Пространство параметров”: фактические параметры вызываемых функций
  - Локальные переменные
  - Сохраненные регистры
  - Прежнее значение указателя фрейма
- Фрейм вызывающей функции
  - Адрес возврата
    - Помещается на стек инструкцией call
  - Значения фактических аргументов для текущего вызова



© 2013 МГУ/ВМиК/СП

```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;
}
```

```
%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)
```