# ????

LIFO. Управляется через регистр RSP. Когда программа начинает выполняться, ОС инициализирует регистр RSP адресом последней ячейки памяти в сегменте стека. Размер стека зависит от системы. На Linux x86-64 стек ограничен 2 мегабайтами.

Стек растет от больших адресов к меньшим. При начале стек выровнен по 16-байтовой границе.

**Использование стека**

push добавляет данные в стек. Возможно добавить 16- и 64-разрядный регистр, адрес в памяти 16- и 64-разрядного числа или значение 16- и 32-разрядной константы (32-битная констранта расширяется до 64 бит).

При выполнении инструкции push от значения регистра RSP вычитается размер операнда. А по адресу, который хранится в стеке, помещается значение операнда.

pop получает из стека значение, адрес которого в регистре RSP. Можно сохранить в 16- и 64-разрядный регистр или адрес в памяти 16- и 64-разрядного числа.

```
global _start

section .text
_start:
    mov rdx, 15
    push rdx            ; в стек помещаем содержимое регистра RDX
    pop rdi             ; значение из вершины стека помещаем в регистр RDI
    mov rax, 60
    syscall
```

Сохранение регистров в стек

```
global _start

section .text
_start:
...
    push rdi
    push rdx
```

```
...
    pop rdx
    pop rdi
```

Сохранение флагов состояния

pushfq и popfq (без аргументов) сохраняет/восстанавливает регистр RFLAGS

**Восстановление стека без извлечения данных**

При завершении программы нужно восстановить адрес в RSP. Можно через pop. Можно прибавть нужное значение к RSP

```
global _start

section .text
_start:
    mov rdi, 11
    mov rdx, 33

    push rdi
    push rdx

    add rsp, 16     ; прибавляем к адресу в RSP 16 байт

    mov rax, 60
    syscall
```

**Резервирование пространства в стеке:**

```
global _start

section .text
_start:
    sub rsp, 16  ; резервируем в стеке 16 байт

    ; некоторая работа со стеком

    add rsp, 16     ; восстанавливаем значение стека
    mov rax, 60
    syscall
```

### Косвенная адресация в стеке

Как и в случае с любым другим регистром, в отношении регистра стека RSP можно использовать косвенную адресацию и обращаться к данным в стеке без смещения указателя RSP.

```
global _start

section .text
_start:
    push 12
    push 13
    push 14
    push 15

    mov rdi, [rsp+16]      ; [rsp+16] - адрес значения 13

    add rsp, 32     ; восстанавливаем значение стека

    mov rax, 60
    syscall
```

---

Revision #3
Created 10 November 2025 02:46:02 by Admin
Updated 10 November 2025 05:20:52 by Admin