CALL ...

RET: ADD...

SP > RET

```
int pow(int base, int exp){
    int res = 1;
    for (int i=0; i<exp; i++)
        res *= base;
    return result;
}
```

§ Calling conventions

1. Порядок аргументов
   1.1) Место размещения аргументов
2. Место результата
   2.1) Возврат составных типов
3. Clean up stack / Очистка стека

---

① stdcall (MSVC)

2. cdecl (Unix)
3. fastcall (Internal functions)
4. thiscall (C++ methods) this передается в регистр, в остальном cdecl
5. vectorcall (SIMD, SSE, AVX, 3D Null)

§ cdecl

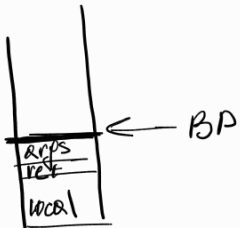
1. Аргументы в стек в обратном порядке

```
int printf(chars format, ...)
```

2. Результат в (основной) регистр
3. Очистка стека от аргументов: вызывающая ф-ция (caller)

§ stdcall

1. Аргумент в стеке в обратном порядке



2. Результат в (основной) регистр

3. Очистка аргументов : в вызываемой ф-ции (callee)

Напишем ф-цию pow (cdecl)

```
FPOW:
PUSH  кол-во лок.   пролог
PUSH  перем.
LD #1
ST &1
CLA
ST &0
```

инициализация

```
FOR: LD &0
CMP &4
BGE EXIT
```

тело

```
PUSH1
PUSH1
LD &5
ST &1
LD &3
ST &0
CALL MUL
SWAP
POP
SWAP
POP
ST &1
LD &0
INC
ST &0
```

exp	4	6
base	3	5
ret	2	4
result	1	3
i	0	2
		1
b'		1
a'		0

```
int pow(int base, int exp) {
    int res = 1;
    for (int i=0; i<exp; i++)
        res *= base;   =mul(res, base);
    return result;
}
```

пролог → тело ф-ции → эпилог
(выделение места) (clean up)

```
int mul(int a, int b);
```

```
SWAP
POP
```

убрали a', b' и сохранили AC

JUMP LO
EXIT: LD &1

SWAP
POP
SWAP
POP
RET

убираем лок. переменные

эпилог

---

stdcall

2 expl 4
1 base 3
0 ret 2
result 1
i 0

FPOW:
PUSH
PUSH
LD #1
ST &1
CLA
ST &0

FOR: LD &0
CMP &4
BGE EXIT

PUSH1
PUSH1
LD &5
ST &1
LD &3
ST &0
CALL MUL
ST &1
LD &0
INC

SWAP
ST &2 - кол-во аргументов
SWAP

ST &0
JUMP L0
EXIT: LD &1

SWAP
POP

SWAP
POP

SWAP
ST &2
SWAP

SWAP
POP

SWAP
POP
RET

Проблема с 32 бит

int32 mul(int32 a, int32 b) могла

b_l
b_h
a_l
a_h

Stack:

А как с int32 в выводе?

1) ret_l
ret_h
b_l
b_h
a_l
a_h

2) зарезерв. ячейки с абсолют. адрес.
типо void mul(int32 a, int32 b).