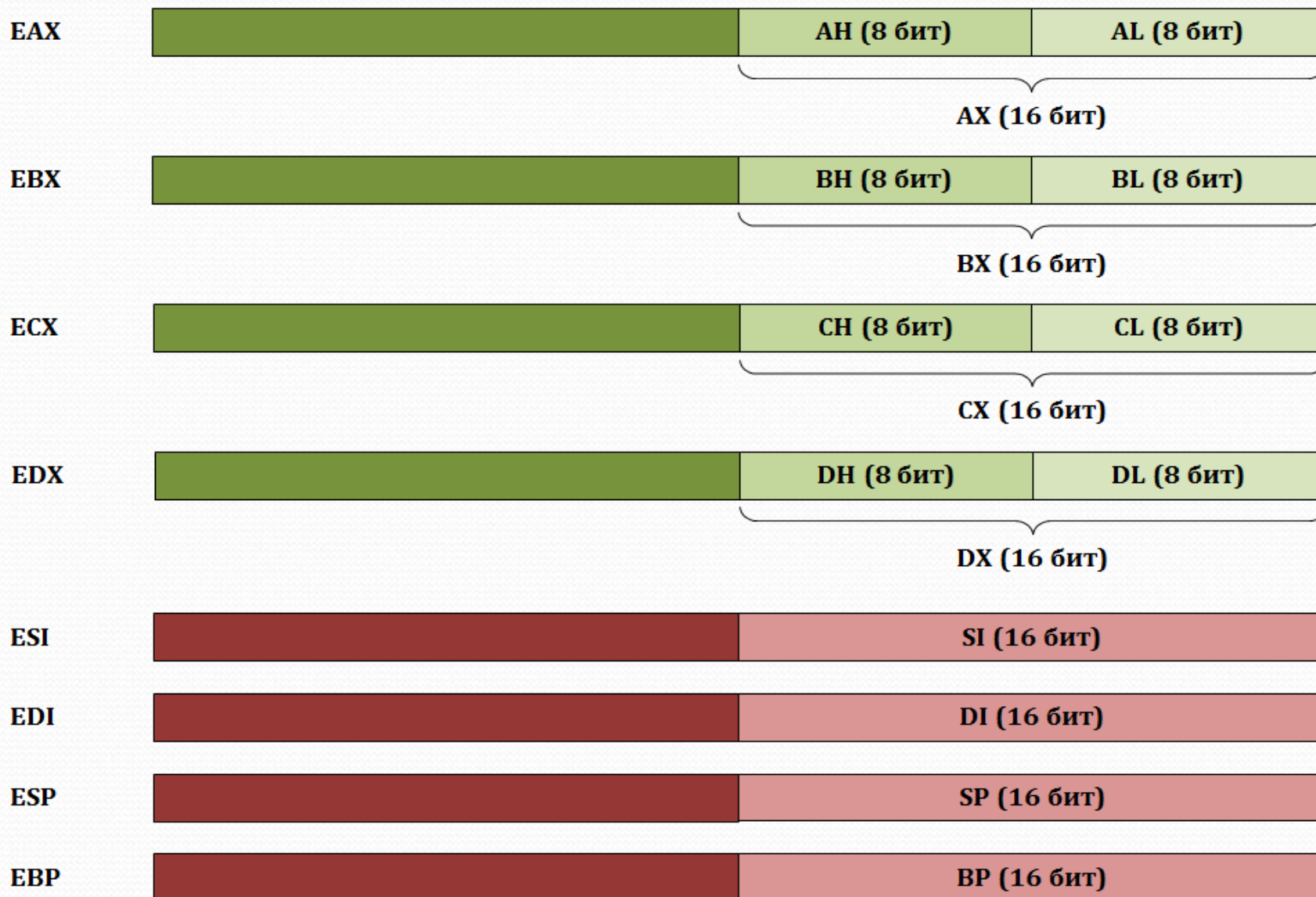


# *Основы языка Assembler*

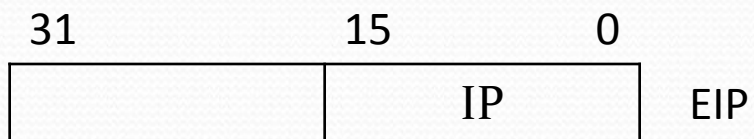
## Связь с программами на других языках

# Регистры общего назначения



## РЕГИСТР ФЛАГОВ

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					OF	DF			SF	ZF		AF		PF		CF



РЕГИСТР СЧЕТЧИК КОМАНД



# Основные понятия языка ассемблера

- Директива эквивалентности позволяет описывать константы:

`<имя> EQU <операнд>`

`A EQU 100`

- Директивы определения данных

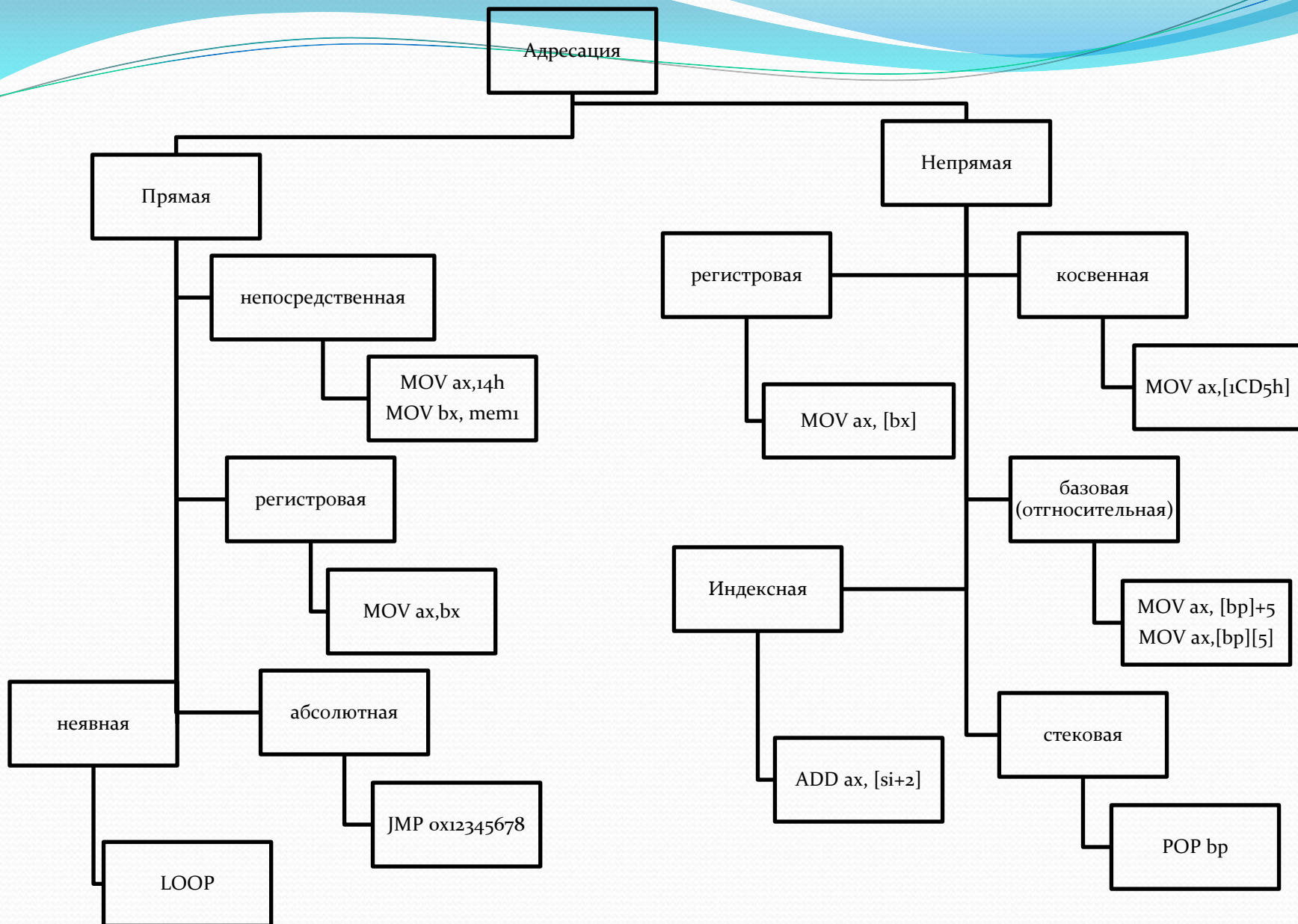
*DB (define byte)* – определяет переменную размером в 1 байт;

`mass db 250 dup (*)`

`a db 10011001b`

`mem DD EC341D7Fh`

- *DW (define word)* – определяет переменную размером в 2 байта (слово);
- *DD (define double word)* – определяет переменную размером в 4 байта (двойное слово);
- *DQ (define quad word)* – определяет переменную размером в 8 байт (четверённое слово);
- *DT (define ten bytes)* – определяет переменную размером в 10 байт.



$$EA = \text{база} + (\text{индекс} * \text{множитель}) + \text{смещение}$$



## Пример прямой адресации

```
mov BP,SP  
mov AX,4C00h
```

```
mov DX, offset mas  
mass db 250 dup ('*')
```

```
mov DL,'!'
```

```
mem1 DW 1D7Fh  
mov ax,mem1
```

```
mem1 DD EC341D7Fh  
mov ax, word ptr mem1 ;ax=1d7f  
mov bx, word ptr mem+2 ;bx=ec34
```

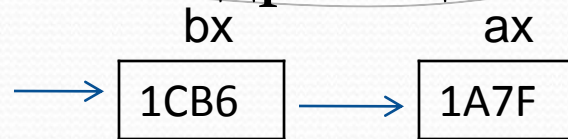
≥

Младший байт		Старший байт	
7F	1D	34	EC

# Пример косвенной адресации

```
mem1 DW 1D7Fh  
lea bx,mem1  
mov ax,[bx]
```

1CB6	1CB7
7F	1A



```
mov DX,[BP]  
mov AL, [DI]
```

```
s1 DB "String 1"  
lea EBX,s1  
mov al, byte ptr [EBX][5] ;al="g"
```

```
mas db 1,2,5,3,7,9,8,3,4  
mov BX,2  
mov DL,mass[BX] ; dl=5
```

Такой же результат даст такая последовательность команд:

```
mov BX,offset mas  
mov DL,2[BX] ; mov DL, [BX+2] mov DL, [BX]+2
```



# Команды передачи данных

## **MOV des,sour**

Примеры:

Mov ax,[numb]

Mov [numb],bx

Mov ax,bx

Mov ax, word ptr [bp+4]

Mov word ptr [bp],14h

## **In reg, port**

## **OUT port, reg**

Примеры:

In ax,dx

In ax,ox60

## **ОШИБКА!!**

Mov [num\_two],[num\_one]

Правильно:

Mov ax,[num\_one]

Mov [num\_two],ax

## **ОШИБКА!!**

Mov ax,ebx

Правильно:

Mov ax,0

Mov ax,bx



## Арифметические команды:

ADD o1,o2 Sub o1,o2  Примеры: Add eax,8 Sub ecx, ebp Add byte [numb],4	INC o1 DEC o1  Примеры: Inc ax Dec dword [numb]	MUL sour DIV sour  Пример: Mov ax,bx Mul cx ;ax=bx*cl  Mov ax,13 Mov cx,3 Div cx ; ax=4 dx=1	IMUL sour IDIV sour  Пример: Imul ecx ;edx:eax=eax*ecx Imul edx,6 ;edx=edx*6 Imul ecx,edx,11 ;ecx=edx*11
------------------------------------------------------------------------------------------	----------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------

## Логические команды:

**AND o1,o2**

**Or o1,o2**

**XOR o1,o2**

**NOT o1**

Пример маскирования:

OR al, 10101010b

# КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Безусловный переход:

**Jmp adr**

...

**L1: mov bx,ax**

**Jmp L1**

Условные переходы:

**Jx adr** переход по равенству флага

**JNx adr** переход по неравенству флага

**Jz adr** ;переход если флаг ZF=1

**Jc adr** ;переход если флаг CF=1

**Js adr** ;переход если флаг SF=1

**Jo adr** ;переход если флаг OF=1

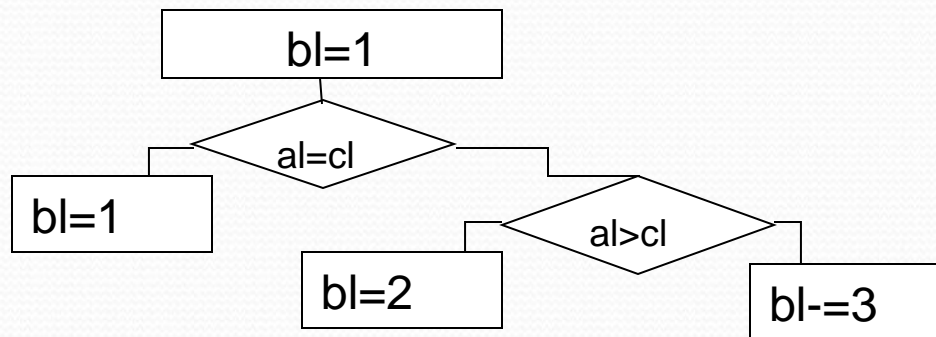


01==02	01!=02 01<>02	01>02	01<02	01<=02	01>=02
JE(JZ)	JNE(JNZ)	JA(JNBE)	JB(JNAE)	JNA(JBE)	JNB(JAE)
Переход если равно	Переход если не равно	Переход если больше	Переход если меньше	Переход если не больше	Переход если не меньше

```

Mov bl,1
Cmp al,cl
Je end_if
Mov bl,2
Cmp al,cl
Ja end_if
Mov bl,3
End_if:

```



Переход по счетчику  
**LOOP metka**

```

start: mov cx,10
for_loop:

```

```

...
    Loop for_loop
final:

```



## *Команды обработки стека:*

**PUSH o1**

**POP o1**

Пример:

Mov ax, 0x1234

Mov bx, 0x5678

Push ax

Push bx

...

Pop bx

Pop ax

## *Команды вызова процедуры и возврата:*

**CALL adr**

**RET**

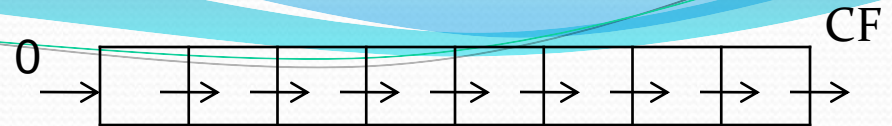
## *Команда загрузки эффективного адреса*

**LEA o1,[o2]**

## Команды сдвига:

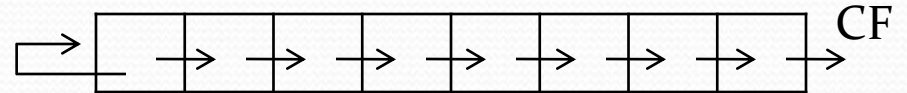
**SHR o1 {,size}**

**SHL o1 {,size}** логический сдвиг числа



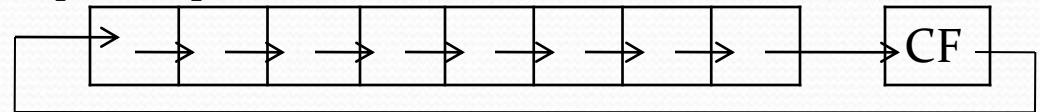
**SAR o1 {,size}**

**SAL o1 {,size}** арифметический сдвиг числа



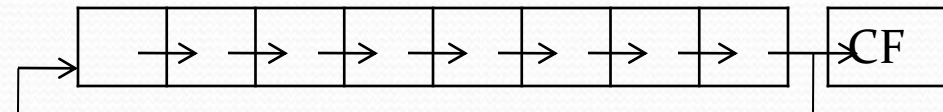
**RCR o1 {,size}**

**RCL o1 {,size}** циклический сдвиг через флаг переноса



**ROR o1 {,size}**

**ROL o1 {,size}** циклический сдвиг с выносом во флаг переноса



ПРИМЕР: Подсчет количества двоичных единиц в числе

Xor bx,bx

Mov cx,16

Repeat:

Shr ax,1

Jnc not\_one

Inc bx

Not\_one: Loop Repeat



*Команды сравнения:*

**CMP 01,02**

Cmp dl,ah

Cmp ax,4

**TEST 01,02**

Test ax,00000100b

*Временное изменение типа переменной:*

**type Ptr выражение**

Mov ax,word ptr [bp+4]

*Псевдокоманды определения констант **DB,DW,DD***

s db 0X55

s1 db 0x55,0x56,'a','hello' ; байтовая последовательность

55	56	61	48	65	6c	6c	6f
----	----	----	----	----	----	----	----

s3 dw 0x1234

;

34	12
----	----

s4 dd 0x12345678

;

78	56	34	12
----	----	----	----





# Каркас программы

.MODEL FLAT, STDCALL

.DATA

<инициализируемые данные>

.DATA?

< неинициализируемые данные>

.CONST

< константы>

.CODE

<метка> proc

< код>

<метка> endp

END

# Интерфейс с языками высокого уровня

## Возврат значений из процедур ассемблера

Размер	Регистр (регистровая пара), в котором возвращается значение
1 байт	AL
2 байта	AX
4 байта	EAX
более 4 байт	процедура должна зарезервировать место для возвращаемого значения и передать его адрес в регистровой паре EDX:EAX.

## Директивы передачи параметров

Директива	Передача параметров	Очистка стека	Использование регистров
fastcall (register)	Слева направо	Процедура	eax,edx,ecx (pascal) ecx,edx (VC++)
pascal	Слева направо	Процедура	нет
cdecl	Справа налево	Вызывающая программа	нет
stdcall	Справа налево	Процедура	нет



## Вход в подпрограмму

Младшие адреса  
←

	ESP	ESP+4	ESP+8
	?	Адрес возврата	Param1
			Param2

void \_stdcall Myproc (int param1,int param2)  
int AddTwo (int,int)

## Выход из подпрограммы

Младшие адреса  
←

		ESP	ESP+4
	?	?	Param1
			Param2

stdcall    ret 8  
C            ret

## stdcall

*имя@nn*  
\_Myproc@8

Myproc(10,15);

push 15; Второй аргумент  
push 10 ; Первый аргумент  
call \_Myproc@8

## C

\_AddTwo

int c=AddTwo(10,15);

push 15; Второй аргумент  
push 10 ; Первый аргумент  
call \_AddTwo  
add esp 8



# Стековый фрейм

Для создания стекового фрейма программа должна выполнить перечисленные ниже действия:

- поместить аргументы в стек;
- вызвать процедуру командой CALL, в результате чего адрес возврата помещается в стек;
- в начале выполнения процедуры сохранить в стеке регистр EBP;
- загрузить в регистр EBP текущий указатель стека из регистра ESP.

## Пример вызова функции

```
extern "C" int razn(int a, int b);  
  
void main()  
{  
    int a,b,c;  
    a=20;  
    b=10;  
    c=razn(a,b);  
    cout << c << "\n";  
}
```

```
.MODEL FLAT, C
```

```
.CODE
```

```
razn proc  a:dword, b:dword  
        ;push ebp  
        ;mov ebp,esp  
        mov eax,word ptr [ebp+8]  
        mov ebx,word ptr [ebp+12]  
        sub eax,ebx  
        ;mov esp,ebp  
        ;pop ebp  
        ret
```

```
razn endp  
end
```

Структура стекового фрейма функции razn

Адрес в SP	Содержимое		
0x0000FFE6	xxxx	Предыдущее значение BP	BP
0x0000FFEA	xxxx	Адрес возврата в main	BP+4
0x0000FFEE	0014	Значение переменной "a"	BP+8
0x0000FFF2	000A	Значение переменной "b"	BP+12



## Пример вызова процедуры

```
extern "C" void sum(int a, int b, int & c);  
void main()  
{  
    int a,b,c;  
    a=10;  
    b=20;  
    sum(a,b,c);  
    cout << c << "\n";  
}
```

Содержимое регистра BP	BP
Адрес возврата в main	BP+4
Значение переменной "a"	BP+8
Значение переменной "b"	BP+12
Адрес переменной "c" в процедуре main	BP+16

```
.MODEL FLAT,C ;  
.CODE  
sum proc a:dword, b:dword, cp:ptr dword  
    push ebp  
    mov ebp,esp  
    push eax  
    push ebx  
    push esi  
    mov eax,dword ptr [ebp+8];  
    mov ebx,dword ptr [ebp+12];  
    add eax,ebx  
    mov esi, dword ptr [ebp+16];  
    mov [esi],eax ;  
    pop esi  
    pop ebx  
    pop eax  
    mov esp,ebp  
    pop ebp  
    ret  
sum endp  
end
```



## Вызов функции с локальными переменными

$$C=(A+B)+(B-A)*B*A$$

```
extern "C" int calc(int a, int b);  
void main()  
{int a,b,c;  
a=10;  
b=20;  
c = calc(a,b);  
cout << c << "\n";  
}
```

Содержимое	
место для 2-ой локальной. переменной	BP-8
место для 1-ой локальной. переменной	BP-4
Содержимое регистра BP	BP
Адрес возврата в main	
Значение переменной "a"	BP+8
Значение переменной "b"	BP+12

```
.MODEL FLAT,C  
.CODE  
calc proc  a:dword, b:dword  
    push ebp  
    mov ebp,esp  
    sub esp,8 ;  
    push ebx  
    mov eax,dword ptr [ebp+8] ; a  
    mov ebx,dword ptr [ebp+12] ; b  
    add eax,ebx  
    mov dword ptr [ebp-4],eax ; a+b  
    mov eax,dword ptr [ebp+8]  
    sub ebx,eax  
    mov dword ptr [ebp-8],ebx ; b-a  
    mov ebx,dword ptr [ebp+12]  
    mul ebx ; a*b  
    mul dword ptr [ebp-8] ; a*b*(b-a)  
    add eax,dword ptr [ebp-4] ; ab(b-a)+(a+b)  
    pop ebx  
    mov esp,ebp  
    pop ebp  
    ret  
calc endp  
end
```