Вызов функции по указателю

```c
typedef int (*arith)(int, int);

int eval(arith pf, int x, int y)
{
   return pf(x, y);
}

int sum(int x, int y) {
   return x + y;
}

int sub(int x, int y) {
   return x - y;
}

int mul(int x, int y) {
   return x * y;
}

int div(int x, int y) {
   return x / y;
}
```

```asm
global sum
sum:
        push    ebp
        mov     ebp, esp
        mov     eax, dword [ebp+12]
        add     eax, dword [ebp+8]
        pop     ebp
        ret
global sub
sub:
        push    ebp
        mov     ebp, esp
        mov     eax, dword [ebp+8]
        sub     eax, dword [ebp+12]
        pop     ebp
        ret
global mul
mul:
        push    ebp
        mov     ebp, esp
        mov     eax, dword [ebp+12]
        imul    eax, dword [ebp+8]
        pop     ebp
        ret
global div
div:
        push    ebp
        mov     ebp, esp
        mov     edx, dword [ebp+8]
        mov     eax, edx   ; cdq
        sar     edx, 31    ;
        idiv    dword [ebp+12]
        pop     ebp
        ret
global eval
eval:
        push    ebp
        mov     ebp, esp
        sub     esp, 8
        mov     edx, dword [ebp+12]
        mov     eax, dword [ebp+16]
        mov     ecx, dword [ebp+8]
        mov     dword [esp], edx
        mov     dword [esp+4], eax
        call    ecx
        mov     esp, ebp
        pop     ebp
        ret
```

```c
#include <stdio.h>

int main() {
   int a = 1, b = 2, c;
   arith pf = sum;
   c = eval(pf, a, b);
   printf("%d\n", c);
   return 0;
}
```

```nasm
%include ‘io.inc’
section .rodata
LC0: db '%d', 10, 0

CEXTERN printf

section .text
global CMAIN
CMAIN:
        lea     ecx, [esp+4]
        and     esp, -16
        push    dword [ecx-4]
        push    ebp
        mov     ebp, esp
        push    ecx
        sub     esp, 20
        mov     dword [esp+8], 2
        mov     dword [esp+4], 1
        mov     dword [esp], sum
        call    eval
        mov     dword [esp+4], eax
        mov     dword [esp], LC0
        call    printf
        mov     eax, 0
        add     esp, 20
        pop     ecx
        pop     ebp
        lea     esp, [ecx-4]
        ret
```

Сравнение строк с помощью функции my_strncmp

```
%include 'io.inc'

BUFSIZE equ 32

section .data
 s1 db 'some text'
    times BUFSIZE-$+s1 db 0

 s2 db 'some text...'
    times BUFSIZE-$+s2 db 0

section .text
global CMAIN
CMAIN:
   push ebp
   mov ebp, esp
   sub esp, 12
   mov dword [esp], s1
   mov dword [esp + 4], s2
   mov dword [esp + 8], BUFSIZE-1
   call my_strncmp ; возвращаем 0,
           ; если строки равны
           ; иначе - номер байта
           ; в котором встретилось
           ; различие (считаем с 1)
   PRINT_DEC 4, eax
   NEWLINE
   xor eax, eax
   mov esp, ebp
   pop ebp
   ret
```

```
my_strncmp:
   push ebp
   mov ebp, esp
   push esi
   push edi
   xor eax, eax
   mov ecx, dword [ebp + 16]
   mov esi, dword [ebp + 8]
   mov edi, dword [ebp + 12]
   cld
   jecxz .end
 .loop:
   cmpsb
   jne .ne
   loop .loop
   jmp .end
 .ne:
   mov eax, dword [ebp + 16]
   sub eax, ecx
   inc eax
 .end:
   pop edi
   pop esi
   mov esp, ebp
   pop ebp
   ret
```

Сравнение строк – приведение к стандартному поведению функции strncmp. Данный код будет работать только если параметр n меньше или равен реальной длины строки!

```
strncmp:
   push ebp
   mov ebp, esp
   push esi
   push edi
   xor eax, eax
   mov ecx, dword [ebp + 16]
   mov esi, dword [ebp + 8]
   mov edi, dword [ebp + 12]
   cld
   repe cmpsb
   je .end
   mov ecx, -1
   mov eax, 1
   cmovb eax, ecx
.end:
   pop edi
   pop esi
   mov esp, ebp
   pop ebp
   ret
```

strnlen

```
#include <string.h>
size_t strnlen (const char *s,
size_t maxlen);
```

```
strnlen:
   push ebp
   mov ebp, esp
   push edi
   xor eax, eax
   mov ecx, dword [ebp + 12]
   mov edi, dword [ebp + 8]
   repne scasb
   mov eax, dword [ebp +12]
   sub eax, ecx
   dec eax
   pop edi
   mov esp, ebp
   pop ebp
   ret
```

memset

```
#include <string.h>
void *memset  (void *s, int c,
size_t n);
```

```
memset:
   push ebp
   mov ebp, esp
   push edi
   mov ecx, dword [ebp + 16]
   mov esi, dword [ebp + 8]
   mov al,  byte  [ebp + 12]
   rep stosb
   pop edi
   mov esp, ebp
   pop ebp
   ret
```