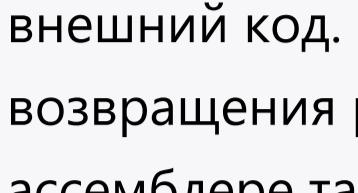


Результат процедуры

Последнее обновление: 02.07.2023



Нередко процедуры выполняют некоторые вычисления и генерируют некоторый результат этих вычислений. Вполне вероятно, что мы захотим передать этот результат во внешний код. Во многих языках программирования высокого уровня есть механизм возвращения результата функции - обычно для этого применяется оператор **return**. В ассемблере такого механизма нет. Наиболее удобным местом для возврата результатов процедуры в архитектуре x86-64 являются регистры.

Как правило, результат процедуры помещается в регистр RAX (или его подрегистры EAX, AX, AL), хотя в реальности, если мы сами пишем всю программу на ассемблере, мы можем использовать для этой цели любой регистр общего назначения. Однако на регистр RAX завязано много соглашений. Так, именно в RAX большинство языков высокого уровня помещают результат функции. А согласно интерфейсу Microsoft ABI при взаимодействии с WinAPI целочисленный результат помещается в регистр RAX, а если результат представляет число с плавающей точкой - то в регистр XMM0.

Если нужно вернуть результат, размер которого превышает 64 бита (например, 128 бит или 256 бит) и соответственно не помещается в RAX или в другой 64-разрядный регистр общего назначения, то можно разделить результат на части и вернуть эти части в двух или более регистрах. Часто можно увидеть, как функции возвращают 128-битные значения в паре регистров RDX:RAX. Конечно, регистры XMM/YMM — еще одно хорошее место для возврата больших значений.

Если надо вернуть большой объект, например, массив из 1000 элементов, то, очевидно, регистры не подходят. И в этом случае можно вместо значения возвратить его адрес (который занимает 8 байт).

Простейший пример возвращения результата:

```
1 .code
2 ; Процедура sum выполняет сложение двух чисел
3 ; RCX - первое число
4 ; RDX - второе число
5 ; RAX - результат сложения
6 sum proc
7     mov rax, rcx
8     add rax, rdx
9     ret
10 sum endp
11
12 main proc
13     ; устанавливаем параметры для процедуры sum
14     mov rcx, 5
15     mov rdx, 6
16     call sum    ; после вызова в RAX - результат сложения
17     add rax, 4  ; далее мы можем использовать этот результат
18     ret
19 main endp
20 end
```

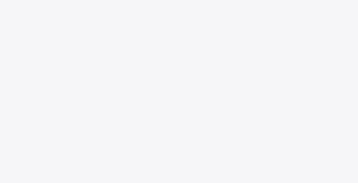
Здесь процедура sum принимает через регистры RCX и RDX два параметра - два числа, складывает их и помещает результат в регистр RAX. После вызова процедуры sum мы можем использовать полученный результат в регистре RAX каким-нибудь образом, например, прибавить к нему еще число.

Другой пример найдем длину массива и поместим результат в RAX:

```
1 .data
2     nums1 word 10, 20, 30, 15, 15
3     nums1Len = $-nums1
4     numSize = 2      ; размер типа word
5 .code
6 ; Процедура sum выполняет сложение чисел массива
7 ; RCX - адрес массива
8 ; RDX - длина массива - количество элементов
9 ; RAX - результат сложения
10 sum proc
11     mov rdi, 0
12     xor rax, rax    ; обнуляем RAX
13     while_begin:
14         cmp rdi, rdx    ; сравниваем количество обработанных элементов
15         je while_end    ; если все элементы обработали, выходим из цикла
16         add ax, word ptr [rcx + rdi*numSize]    ; складываем значение по адресу [rcx
17         inc rdi        ; инкрементируем счетчик обработанных элементов
18         jmp while_begin ; переходим к обработке нового элемента массива
19     while_end:
20         ret
21     sum endp
22
23 main proc
24     ; устанавливаем параметры для процедуры sum
25     lea rcx, nums1           ; в RCX адрес массива
26     mov rdx, nums1Len / numSize ; в RDX - количество элементов массива
27     call sum    ; после вызова в RAX - результат сложения
28     ret
29 main endp
30 end
```

Здесь функция sum принимает адрес массива и количество элементов в массиве. В цикле проходит по всем элементам массива, используя косвенную адресацию и счетчик элементов - регистр RDI и помещает в регистр RAX сумму элементов массива.

[Назад](#) [Содержание](#) [Вперед](#)



Помощь сайту

[Помощь сайту](#)

Юмани:

410011174743222

Номер карты:

4048415020898850

[Телеграмм](#)

[Вконтакте](#) | [Телеграм](#) | [Донаты/Помощь сайту](#)

Contacts: metanit22@mail.ru

Copyright © Евгений Попов, metanit.com, 2026. Все права защищены.