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:

```
.code
; Процедура sum выполняет сложение двух чисел
; RCX - первое число
; RDX - второе число
; RAX - результат сложения
sum proc
    mov rax, rcx
    add rax, rdx
    ret
sum endp

main proc
    ; устанавливаем параметры для процедуры sum
    mov rcx, 5
    mov rdx, 6
    call sum    ; после вызова в RAX - результат сложения
    add rax, 4  ; далее мы можем использовать этот результат
    ret
main endp
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:

```
.data
    nums1 word 10, 20, 30, 15, 15
    nums1Len = $-nums1
    numSize = 2     ; размер типа word
.code
; Процедура sum выполняет сложение чисел массива
; RCX - адрес массива
; RDX - длина массива - количество элементов
; RAX - результат сложения
sum proc
    mov rdi, 0
    xor rax, rax    ; обнуляем RAX
while_begin:
    cmp rdi, rdx    ; сравниваем количество обработанных элементов
    je while_end    ; если все элементы обработали, выходим из цикла
    add ax, word ptr [rcx + rdi*numSize]    ; складываем значение по адресу [rcx
    inc rdi         ; инкрементируем счетчик обработанных элементов
    jmp while_begin ; переходим к обработке нового элемента массива
while_end:
    ret
sum endp

main proc
    ; устанавливаем параметры для процедуры sum
    lea rcx, nums1                  ; в RCX адрес массива
    mov rdx, nums1Len / numSize     ; в RDX - количество элементов массива
    call sum    ; после вызова в RAX - результат сложения
    ret
main endp
end
```

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

Назад **Содержание** **Вперед**