Макросы

Определение макросов

Последнее обновление: 03.07.2023

В ассемблере макрос представляет идентификатор, который ассемблер преобразует в дополнительный код. Макросы позволяют заменить длинные повторяющиеся последовательности код гораздо более короткими последовательностями. В каком-то смысле макросы MASM можно назвать процедурами/функциями времени компиляции.

Определение макроса в общем случае имеет следующую форму:

```
имя_макроса macro аргументы_макроса
    код макроса
endm
```

Сначала идет объявление макроса - имя макроса, затем оператор `macro` и после указываются аргументы макроса. Аргументы необязательны, макрос может вообще не иметь аргументов. Завершается макрос оператором `endm`. Между объявлением макроса и `endm` располагается собственно код макроса.

Рассмотрим пример макроса, который складывает 128-разрядное число с 64-разрядным:

```
.data
    num qword 0ffffffffffffffffh,   ; младшие 64 бит
                                4   ; старшие 64 бит
.code
add128 macro
    add rax, rbx
    adc rdx, 0
endm

main proc
    mov rax, num        ; младшие 64 бит - в RAX
    mov rdx, num[8]     ; старшие 64 бит - в RDX
    mov rbx, 3
    add128              ; вставляем макрос
    ret
main endp
end
```

В данном случае макрос называется `add128`:

```
add128 macro
    add rax, rbx
    adc rdx, 0
endm
```

Здесь подразумеваем, что 128-разрядное число располагается в регистрах RDX:RAX: младшие 64 битов в RAX, а старшие 64 битов - в RDX. Прибавляемое 64-битное число передается через регистр RBX. Работа макроса проста - складываем 64-битное число RBX с младшими 64 битами RAX и, если при этом устанавливается флаг переноса, прибавляем бит к старшим 64-битам в RDX.

Здесь 128-битное число представляет массив num из двух подчисел qword, которое загружается в RAX и RDX. Для вставки макроса в код применяется имя макроса. То есть макрос не вызывается в процессе выполнения, его код вставляется в место вызова макроса на этапе компиляции. То есть фактически мы получим программу:

```
.data
    num qword 0ffffffffffffffffh,   ; младшие 64 бит
                                4   ; старшие 64 бит
.code
main proc
    mov rax, num        ; младшие 64 бит - в RAX
    mov rdx, num[8]     ; старшие 64 бит - в RDX
    mov rbx, 3
    add rax, rbx
    adc rdx, 0
    ret
main endp
end
```

Макросы или функции?

В приведенном выше примере мы могли бы определить макрос сложения чисел в виде отдельной процедуры. И тут может возникнуть вопрос: что выбрать - макросы или функции? Минусом макросов является то, что их код вставляется в каждое место, где они используются. Что ведет к общему увеличению объема программы. С другой стороны, макросы позволяют избежать переходов и сохранений/восстановлений адреса следующей инструкции, что положительно влияет на производительность. Макросы немного быстрее, чем вызовы процедур, потому что вы не выполняете вызов и соответствующие инструкции ret. Кроме того, с макросами сам код становится чуть более читабельным. В общем случае рекомендуется применять макросы для коротких, критичных по времени частей программы. Тогда как процедуры применяются для более длинных блоков кода и когда время выполнения не так критично.

Назад **Содержание** **Вперед**