Документация по языку Lumin

**НАЧАЛО**

* **Компиляция программы**

Для компиляции вашей программы откройте файл Lumin.exe и введите [comp путь к файлу для компиляции], когда она завершится, то файл с ассемблерным кодом и выходной файл появятся в той же директории что и файл с кодом.

**СПИСОК КОМАНД**

* **import**

Импортирует ассемблерный файл в программу

Можно импортировать файлы Lumin’а

Пример:

import “lib/std.asm”

* **inter**

Вызывает прерывание

Пример:

inter 20h

* **delete**

Удаляет стековою переменную

Пример:

delete ([имя стековой переменной для удаления],[количество байт для удаления])

* **defbyte**

Определяет байт без возможности к нему обратится

Пример:

defbyte [значение]

* **format**

Устанавливает формат для программы

Пример:

format PE Console

Cписок всех форматов:

PE Console

PE64 Console

PE GUI

PE64 GUI

binary

coff

ms coff

MZ

* **dllimport**

Импортирует функции из dll-библиотек

Пример:

dllimport “[название библиотеки]”,[функции…]

* **sector**

Указывает сектор

Пример:

sector [номер сектора]

* **reserve**

Резервирует память указанным типом данных и указанное количество раз

Пример:

reserve [тип данных],[сколько раз]

* **savestack**

Сохраняет значение в стек

Пример:

savestack [регистр или значение]

* **restorestack**

восстанавливает значение из стека

Пример:

restorestack [регистр или значение]

* **locallabel**

Определяет локальную метку для макроса или процедуры

Пример:

locallabel [название метки]

label [название метки]:

* **return**

Возвращает 32 битное значение (в eax) и делает возврат из процедуры или метки

Пример:

return [значение]

* **crttemp**

Создаёт экземпляр шаблона

Пример:

crttemp [название исходного шаблона],[название нового экземпляра шаблона](параметры для конструктора)

* **entrypoint**

Определяет точку входа в программу

Пример:

entrypoint main

* **goto**

Переходит к метке

Пример:

goto [название метки]

* **define**

Определяет значение для замены на другое значение

Пример:

define [на что],[что]

* **undefine**

Перестаёт определять значение

Пример:

undefined [какое значение перестать определять]

* **stop**

Останавливает выполнение программы не завершая её

* **compare**

сравнивает два значения, где первый операнд это регистр или переменная, и второй операнд любое значение

Пример:

compare [переменная или регистр],[значение]

Также следующие команды выполняют переход к метке по их условию:

ifgreaterequ

ifgreater

iflessequ

ifless

ifnotequ

ifequ

ifzero

ifnotzero

Пример:

compare variable,1

ifequ golabelequ

**ПРОЦЕДУРЫ, МАКРОСЫ, СТРУКТУРЫ И МЕТКИ**

* **Определение метки**

**Чтобы определить метку напишите:**

**label [название метки]:**

* **Действия с меткой**

**Чтобы вызвать метку напишите:**

**[название метки]()**

**Чтобы перейти к метке напишите:**

**goto [название метки]**

* **Определение процедуры**

**Чтобы определить процедуру напишите:**

**proc [название процедуры]([параметры если надо])**

**{**

**}**

**Пример:**

**Proc main(args:byte)**

**{**

**}**

**Действия с процедурой**

**Чтобы вызвать метку напишите:**

**[название процедуры]([параметры])**

**Чтобы перейти к метке напишите:**

**goto [название процедуры]**

* **Макросы**

**Чтобы определить макрос напишите:**

**macro [название процедуры]([параметры если надо])**

**{**

**}**

**Чтобы вызвать метку напишите:**

**[название процедуры]([параметры])**

* **Структуры**

**Чтобы определить структуру напишите:**

**struct [название структуры]([параметры для присвоения переменным])**

**{**

**}**

**Обращение к переменным в которые находятся в структуре через точку**

**ЦИКЛЫ И IF**

\*Используйте not() или ~() для отрицания и break для выхода из цикла

* **IF**

Пример:

if a==a

{

}

* **WHILE**

Пример:

while a==a

{

}

* **DOWHILE**

Пример:

dowhile a==a

{

}

* **FOR**

Пример:

for [переменная для буфера],[до скольки идти],[шаг]

{

}

* **FOREACH**

Пример:

foreach [переменная для буфера],[массив],[сколько элементов в массиве]

{

}

**ДЕЙСТИЯ С ПЕРЕМЕННЫМИ**

[название переменной] += 2 - плюс

или ++

[название переменной] -= 2 - минус

или --

[название переменной] \* 2 - умножение

[название переменной] = 2 - равенство

[название переменной] / [ число] - деление

<< для байтового сдвига на лево, на право - >>

пример:

eax@ << cl@

**УКАЗАТЕЛИ**

Чтобы использовать указатели в конце переменной поставьте @

Пример:

var1 = var2@

**ТИПЫ ДАННЫХ**

byte - определяет один байт данных

word - определяет двухбайтовое слово (16 бит)

ubyte - определяет двухбайтовое слово в unicode (16 бит)

dword - определяет четырехбайтовое слово (32 бита)

qword - определяет восьмибайтовое слово (64 бита)

tword - определяет десятибайтовую последовательность (80 бит)

Пример создания переменной:

byte [название переменной] = 0

Для создания массива просто пишите значения через запятую:

dword [название переменной] = 0,9,6,34

Также можно создавать стековые переменные которых значение находится на стеке:

stack byte [название переменной] = [значения],[сколько байтов /

]

**ПРИМЕЧАНИЯ**

Каждая программа должна иметь процедуру main

Можно писать со скобочками и без

Можно использовать ; для разделения команд

mode64 и mode16 для изменения битности программы

“строка”@ означает что будет создан байтовый массив

# для комментариев