Другие темы раздела

FASM Fasm dll https://www.cyberforum.ru/ fasm/ thread1252260.html
Как в fasm создать dll файл?

FASM Как использовать структуру sqlite3?
Хотелось бы прикрутить sqlite к своей проге с dll кой проблем нет. где взять структуры описанные в sqlite3.c

Делаем в IDE FASM'а кнопку Debug и дружим его с OllyDbg FASM
Делаем в IDE FASM'а кнопку Debug и дружим его с OllyDbg статья была взята здесь Статья посвящена всем любителям компилятора FASM и тем кто пишет код используя его IDE. Известно что любое...

FASM Как вывести время работы программы? Доброе время, суток! У меня такой вопрос, как вывести время работы программы? Скажем есть такая простенькая программа, которая считает от 1 миллиарда до 0, вот как сделать чтоб после того как она... https://www.cyberforum.ru/ fasm/ thread1248143.html

FASM Бегущая строка в текстмоде, нежно. Насилуем знакогенератор https://www.cyberforum.ru/ fasm/ thread1244262.html
Вотъ. Использованы куски из моей XVGA, писаные ещё в 1992, так что не обессудьте. В качестве мана пользовал Richard Wilton, "Programmer's Guide to PC and PS/2 Video Programming." ;FASM - сохранять...

Мануал по flat assembler FASM
flat assembler 1.71 Мануал программера перевод "flat assembler 1.71 Programmer's Manual" by Tomasz Grysztar перевод выполнили Paranoik и Mikl___ Содержание ...

FASM Уроки Iczelion'a на FASM
Уроки Iczelion'a на FASM Урок первый. MessageBox на FASM format PE GUI include 'win32ax.inc' ; import data in the same section invoke MessageBox,NULL,msgBoxText,msgBoxCaption,MB_OK ...

FASM Побайтовый вывод файла Пытаюсь ввести в консоль файл в шестнадцатеричном виде, но происходит ошибка при выполнении. format PE console 4.0 include 'win32a.inc' xor ebx, ebx ; invoke CreateFile,\ ... https://www.cyberforum.ru/ fasm/ thread1219549.html

FASM Вывод адреса на консоль https://www.cyberforum.ru/ fasm/ thread1219432.html
Пытаюсь на консоль вывести адрес fin: invoke printf, не робит - как правильно надо? format PE console 4.0 entry start include 'win32a.inc' section '.data' data readable fin ...

FASM ГСЧ на макросах Всем привет. Понадобилось заюзать ГСЧ посредством макросов, чтобы каждый раз на стадии компиляции, использовалось уникальное значение. Учитывая семантику препроцессора (там чёрт ногу сломит),... https://www.cyberforum.ru/ fasm/ thread1213146.html

Mikl___
Ушел с форума
13980 / 6996 / **810**
Регистрация: 11.11.2010
Сообщений: 12,580

09.09.2014, 12:54 **[ТС]** **0**

Руководство по препроцессору FASM

09.09.2014, 12:54. **Просмотров** 10863. **Ответов** 7
Метки (Все метки)

Ответ

1. Об этом документе

Я написал это потому что вижу, как многие задают вопросы на форуме FASM, связанные с непониманием идей или особенностей препроцессора. (Я не отговариваю Вас задавать такие вопросы, непонимание чего-то - это вполне нормально, и если Ваш вопрос не чересчур сложен, кто-нибудь наверняка на него ответит).

Если Вам что-нибудь из туториала покажется непонятным, пожалуйста, напишите на форум FASM, форум WASM, автору или переводчику.

2. Общие понятия

2.1. Что такое препроцессор

Препроцессор - это программа (или чаще - часть компилятора), которая преобразует исходный текст непосредственно перед компиляцией. К примеру, если Вы используете какой-либо кусок кода довольно часто, можно дать ему некое имя и заставить препроцессор повсеместно заменять это имя в исходном тексте на соответствующий ему код.

Другой пример - Вы хотите имитировать инструкцию, которая на самом деле не существует. В таком случае препроцессор может заменять её последовательностью инструкций дающих желаемый эффект.

Препроцессор просматривает исходный текст и заменяет некоторые вещи другими. Но как объяснить препроцессору, что именно он должен делать? Для этих целей существуют директивы препроцессора. О них мы и будем говорить.

Препроцессор понятия не имеет о инструкциях, директивах компилятора и прочих подобных вещах. Для него существуют собственные команды, и он игнорирует всё остальное.

2.2. Комментарии ";"

Подобно большинству ассемблеров, комментарии в FASM начинаются с точки с запятой ";". Всё, что следует за этим символом до конца строки игнорируется и удаляется из исходника.

К примеру, исходный текст

Assembler Выделить код

Assembler Выделить код

```
1  ; заполним 100h байтов адресуемых EDI нулями
2  xor eax, eax    ; обнуляем eax
3  mov ecx, 100h/4
4  rep stosd
```

после препроцессора превращается в

Assembler Выделить код

```
1  xor eax,eax
2  mov ecx,100h/4
3  rep stosd
```

ПРИМЕЧАНИЕ: **;** можно рассматривать как директиву препроцессора, удаляющую текст начиная с этого символа до конца строки.

ПРИМЕЧАНИЕ: Строка, полностью состоящая из комментария не будет удалена. Она становится пустой строкой (см. пример выше). Это будет важно в дальнейшем.

2.3. Перенос строки (Line Break "\")

Если строка выглядит слишком длинной, возможно разделить её на несколько, используя символ "****". При обработке препроцессором следующая строка будет добавлена к текущей.

Например:

Assembler Выделить код

```
1  db  1, 2, 3,\
2      4, 5, 6,\
3      7, 8, 9
```

будет преобразовано в:

Assembler Выделить код

```
1  db  1,2,3,4,5,6,7,8,9
```

Конечно, **** в составе текстовой строки или комментария не вызовет объединения строк. Внутри текстовой строки этот символ воспринимается как обычный ASCII символ (как и всё остальное заключённое между кавычками **'** или **"**). Комментарии же удаляются без анализа того, что в них написано.

В строке после символа **** могут быть только пробелы или комментарии.

Ранее, я упоминал, что строка, состоящая только из комментария не удаляется, а заменяется на пустую строку. Это значит, что код, подобный этому:

Assembler Выделить код

```
1  db  1, 2, 3,\
2  ;   4,5,6,\   - закомментировано
3      7, 8, 9
```

преобразуется в:

Assembler Выделить код

```
1  db  1, 2, 3
2      7, 8, 9
```

и вызовет ошибку. Выход из положения - помещать символ **** до комментария:

Assembler Выделить код

```
1  db  1, 2, 3,\
2  \;  4,5,6     - правильно закомментировано
3      7, 8, 9
```

в результате будет:

Assembler Выделить код

```
1  db  1, 2, 3, 7, 8, 9
```

как мы и хотели.

2.4. Директива *INCLUDE*

Синтаксис:

Assembler Выделить код

```
1  include <некая строка содержащая имя файла file_name>
```

Эта директива вставляет содержимое файла **file_name** в исходный текст. Вставленный текст, естественно, тоже будет обработан препроцессором. Имя файла (и путь к нему, если он есть) должны быть заключены в кавычки **"** или апострофы **'**.
Например:

Assembler Выделить код

```
1  include 'file.asm'
2  include 'HEADERS\data.inc'
3  include '..\lib\strings.asm'
4  include 'C:\config.sys'
```

Можно также использовать переменные окружения ОС, помещая их имена между символами **%**:

Assembler Выделить код

```
1  include '%FASMINC%\win32a.inc'
2  include '%SYSTEMROOT%\somefile.inc'
3  include '%myproject%\headers\something.inc'
4  include 'C:\%myprojectdir%\headers\something.inc'
```

2.5. Strings preprocessing

You may have problem to include ' in string declared using 's or " in string declared using "s. For this reason you must place the character twice into string, in that case it won't end string and begin next as you may think, but it will include character into string literaly.

For example:

Assembler Выделить код

```
1  db 'It''s okay'
```

will generate binary containing string **It's okay**.
It's same for **"**.

3. Присваивания (Equates)

3.1. Директива *EQU*

Простейшая команда препроцессора.

Синтаксис:

Assembler Выделить код

```
1  <name1> equ <name2>
```

Это команда говорит препроцессору, что необходимо заменить все последующие <**name1**> на <**name2**>.
Например:

Assembler Выделить код

```
1  count   equ 10  ; это команда препроцессора
2  mov ecx, count
```

преобразуется в:

Assembler Выделить код

```
1  mov ecx, 10
```

Ещё пример:

Assembler Выделить код

```
1  mov eax, count
2  count   equ 10
3  mov ecx, count
```

преобразуется в:

Assembler Выделить код

```
1  mov eax, count
2  mov ecx,10
```

потому что препроцессор заменит **count** только после директивы **equ**.
Даже это работает:

Assembler Выделить код

```
1  10  equ 11
2  mov ecx, 10
```

после обработки препроцессором, получим:

Assembler Выделить код

```
1  mov ecx, 11
```

Обратите внимание, **name1** может быть любым идентификатором. Идентификатор - это всего лишь набор символов, завершаемый пробелом (**space**), символом табуляции (**tab**), концом строки (**EOL**), комментарием **;**, символом переноса строки **** или оператором, включая операторы ассемблера и/или специальные символы вроде **,** или **}**.
name2 может быть не только единичным идентификатором, берутся все символы до конца строки. **name2** может и отсутствовать, тогда **name1** будет заменен на пустое место.

Например:

Assembler Выделить код

```
1  10  equ 11, 12, 13
2  db  10
```

получим:

Assembler Выделить код

```
1  db  11, 12, 13
```

3.2. Директива *RESTORE*

Можно заставить препроцессор прекратить заменять идентификаторы, определённые директивой **EQU**. Это делает директива **RESTORE**

Синтаксис:

Assembler Выделить код

```
1  restore <name1>
```

name1 - это идентификатор определённый ранее в директиве **EQU**. После этой команды **name1** больше не будет заменяться на **name2**.
Например:

Assembler Выделить код

```
1  mov eax, count
2  count   equ 10
3  mov eax, count
4  restore count
5  mov eax, count
```

получим:

Assembler Выделить код

```
1  mov eax, count
2  mov eax, 10
3  mov eax, count
```

Обратите внимание, что для определений сделанных директивой **EQU** работает принцип стека. То есть, если мы два раза определим один и тот же идентификатор используя **EQU**, то после однократного использования **RESTORE** значение идентификатора будет соответствовать определённому первой директивой **EQU**.
Например:

Assembler Выделить код

```
1   mov eax, count
2   count   equ 1
3   mov eax, count
4   count   equ 2
5   mov eax, count
6   count   equ 3
7   mov eax, count
8   restore count
9   mov eax, count
10  restore count
11  mov eax,count
12  restore count
13  mov eax,count
```

получим:

Assembler Выделить код

Assembler Выделить код

```
1  mov eax, count
2  mov eax, 1
3  mov eax, 2
4  mov eax, 3
5  mov eax, 2
6  mov eax, 1
7  mov eax, count
```

Если попытаться выполнить **RESTORE** большее количество раз, чем было сделано **EQU**, никаких предупреждений выдано не будет. Значение идентификатора будет неопределенно.
Например:

Assembler Выделить код

```
1  mov eax, count
2  restore count
3  mov eax, count
```

получим:

Assembler Выделить код

```
1  mov eax, count
2  mov eax, count
```

Вернуться к обсуждению:
Руководство по препроцессору FASM

Следующий ответ

2

IT_Exp
Эксперт
87844 / 49110 / **22898**
Регистрация: 17.06.2006
Сообщений: 92,604

09.09.2014, 12:54

Требуется директива препроцессору
Создаю проект "Консольное приложение" на Visual C#. Код : #include <stdio.h> int main(void) {...

Видимость переменных и директивы препроцессору, не видит поле
Есть поле public float zoomSpeed = 0; Есть метод, в нем строки для разных платформ. void...

При создании файла заголовка в Code::Blocks вставляются какие-то команды препроцессору.
Вот что появляется при создании файла rectangle.hpp: #ifndef RECTANGLE_HPP_INCLUDED #define...

Руководство
Как вообще по spring 4 его руководство читать, может кто-нибудь переведет или че путное есть а не...

0