Другие темы раздела

FASM Уроки Iczelion'a на FASM https://www.cyberforum.ru/ fasm/thread1240590.html
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 ...

Побайтовый вывод файла FASM
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,\ ...

FASM Вывод адреса на консоль
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

FASM Создание окна на fasm https://www.cyberforum.ru/ fasm/thread1209394.html
Всем привет. Только что начал изучать ассемблер fasm. Возник первый вопрос: как создать окно? Прошу не просто дать мне код, а ещё объяснить что значит. Заранее благодарен

FASM Вызываем функции из clib (библиотека Си) в DOS
Вобщем, сбылась мечта идиота. Теперь, нежели писать свой ввод/вывод(особенно всегда напрягал ввод/вывод вещественных чисел на экран), можно воспользоваться стандартными ф-циями из библиотеки языка...

Организовать вычисления по формуле FASM
привет, всем активным участникам этого чудесного форума!!! помогите, пожалуйста, написать программу на Fasm Assembler. задание: Создать программу на языке Ассемблер, что позволяет организовать...

FASM Как сделать выход по ESC org 100h old dw 0 jmp start number dw 0 c dw 0 start: xor ax,ax mov es,ax cli https://www.cyberforum.ru/ fasm/thread1161834.html

FASM Получение CLSID image/png https://www.cyberforum.ru/ fasm/ thread1160365.html
Всем ку! int GetEncoderClsid(const WCHAR* format, CLSID* pClsid) { UINT num = 0; // number of image encoders UINT size = 0; // size of the image encoder array in bytes...

FASM Вывод трех строк в один MessageBox Здравствуйте, помогите, пожалуйста, с такой проблемой: не могу вывести 3 строки (Год+Месяц+День) в один MessageBox Вот такой код: format PE GUI 4.0 entry start include 'win32ax.inc' include... https://www.cyberforum.ru/ fasm/thread1142589.html

MikI
Ушел с форума
13987 / 7000 / **813**
Регистрация: 11.11.2010
Сообщений: 12,592

01.09.2014, 06:02 **[TC]** **0**

Мануал по flat assembler

01.09.2014, 06:02. **Просмотров** 99783. **Ответов** 50
Метки (Все метки)

Ответ

Глава 3. Программирование для Windows

Версия FASM'a для Windows включает пакет стандартных файлов разработанных, чтобы помочь создавать программы для Windows. Этот пакет содержит заголовочные файлы для программирования под Win32/Win64 находящиеся в корневой папке а также специализированные файлы находящиеся в поддиректориях. Вообще, заголовочные файлы Win32 включают все требуемые специализированные файлы для Вас, хотя иногда Вы могли бы предпочесть включать часть макросов самостоятельно (так как некоторые из них не включены в заголовочные файлы). Есть двенадцать заголовочных файлов Win32/Win64, которые Вы можете выбрать , с названиями, начинающимися с "win32/win64", и продолжающихся символом "a" чтобы использовать кодирование ASCII, или символом "w" для Unicode.

win32a.inc, win32w.inc, win64a.inc и win64w.inc — основные заголовочные файлы, win32ax.inc, win32wx.inc, win64ax.inc и win64wx.inc — расширенные заголовочные файлы, они содержат более расширенные макросы, эти расширения будут обсуждены отдельно. Наконец win32axp.inc, win32wxp.inc, win64axp.inc и win64wxp.inc — те же самые расширенные заголовочные файлы с особенностью проверки параметров при вызовах процедур.

Вы можете включить заголовочные файлы, любым путем которые предпочитаете, указывая полный путь или используя переменную окружения, но самый простой метод состоит в том, чтобы определить переменную окружения **INCLUDE**, должным образом указывающую на каталог, содержащий заголовочные файлы и затем включить их например:

Assembler Выделить код

```
1  include 'win32a.inc'
```

Важно обратить внимание, что все макросы, в противоположность внутренним директивам FASM, являются чувствительными к регистру, и строчные буквы используются для большинства из них. Если Вы желаете использовать их по другому чем по умолчанию, Вы должны сделать соответствующие корректировки директивой "**fix**".

3.1 Основные заголовочные файлы

Основные заголовочные файлы win32a.inc, win32w.inc, win64a.inc и win64w.inc включают константы и структуры Win32/Win64 и обеспечивают стандартный набор макросов.

3.1.1 Структуры

Все заголовки допускают макрос "**struct**", который позволяет определять структуры более традиционно, подобно другим ассемблерам чем "**struc**" директива. Определение структуры должно быть начато с макроинструкции "**struct**" , сопровождаемой именем, и заканчиваться "**ends**". В строках определения структуры разрешены только метки, являющимися названиями для ее членов:

Assembler Выделить код

```
1  struct POINT
2      x dd ?
3      y dd ?
4  ends
```

Строка с таким определением:

Assembler Выделить код

```
1  point1 POINT
```

объявит структуру **point1** с членами **point1.x** и **point1.y** задавая им значения по умолчанию — те же самые что предусмотрены при определении структуры (в этом случае, значения по умолчанию являются неинициализированными). Но при объявлении структура также принимает, такое же количество параметров сколько и членов в структуре, отменяя значения определенные по умолчанию для этих членов. Например:

Assembler Выделить код

```
1  point2 POINT 10,20
```

инициализирует члена **point2.x** значением 10, и **point2.y** значением 20. "**struct**" макрокоманда не только допускает, объявление структуры данного типа, но также и определяет метки для смещений членов в структуре и константы определяющие размер каждого члена и структуры в целом. Например вышеупомянутое определение структуры **POINT** определяет метки **POINT.x** и **POINT.y**, которые являются смещениями членов в структуре, и **sizeof.POINT.x**, sizeof.POINT.y и **sizeof.POINT** как размеры соответствующих членов и целой структуры. Метки смещения могут использоваться чтобы обратиться к структурам, к которым обращаются косвенно, например:

Assembler Выделить код

```
1  mov eax,[ebx+POINT.x]
```

когда регистр **EBX** содержит указатель на структуру **POINT**. Обратите внимание, также что проверка размера членов будет выполнена.

Сами структуры также могут быть в определениях структуры, так что структуры могут содержать некоторые другие структуры как члены:

Assembler Выделить код

```
1  struct LINE
2      start POINT
3      end POINT
4  ends
```

Когда никакие значения по умолчанию для членов вложенной структуры не определены, как в этом примере, значения по умолчанию берутся из определения вложенной структуры.

Так как значение для каждого члена — отдельный параметр в объявлении структуры, при инициализации вложенной структуры параметры для каждой из них должны быть сгруппированы в отдельную группу например:

Assembler Выделить код

```
1  line1 LINE <0,0>,<100,100>
```

Вышеупомянутое объявление инициализирует каждого члена из **line1.start.x** и **line1.start.y** 0, и **line1.end.x** и **line1.end.y** 100.

Когда структура инициализируется данными меньшими чем размер соответствующего члена, она дополняется до размера неопределенными байтами (а когда большими, случается ошибка). Например:

Assembler Выделить код

```
1  struct FOO
2      data db 256 dup (?)
3  ends
```

Assembler Выделить код

```
1  some FOO <"ABC",0>
```

заполняет первые четыре байта **some.data** определенными значениями и резервирует остальное.

В структурах также могут быть определены "**union**" — объединения а также безымянные вложенные структуры. Определение объединения должно начаться с "**union**" и оканчиваться "**ends**", например:

Assembler Выделить код

Assembler Выделить код

```
1  struct BAR
2      field_1 dd ?
3      union
4          field_2 dd ?
5          field_2b db ?
6      ends
7  ends
```

Каждый из членов, определенных в объединении имеет то же самое смещение, и они совместно используют одну и ту же область памяти. Только первый член объединения инициализируется данным значением, значения для остальных членов игнорируются (однако, если один из других членов требует большего количества памяти чем первый, объединение дополняется до требуемого размера неопределенными байтами). Целое объединение будет инициализировано единственным параметром, данным в объявлении структуры, и этот параметр дает значение первому члену объединения.

Безымянные вложенные структуры объявляются также как "**union**", только начинаются со строки "**struct**" например:

Assembler Выделить код

```
1  struct WBB
2      word dw ?
3      struct
4          byte1 db ?
5          byte2 db ?
6      ends
7  ends
```

Такая вложенная структура принимает только один параметр при объявлении структуры, чтобы инициализировать ее значения, этим параметром может быть группа параметров, определяющих каждого члена вложенной структуры. Так что вышеупомянутый тип структуры можно объявить так:

Assembler Выделить код

```
1  my WBB 1,<2,3>
```

К членам объединений и безымянных вложенных структур обращаются так же, как если бы они были непосредственно члены родительской структуры. Например с вышеупомянутым объявлением **my.byte1** и **my.byte2** — являются правильными метками для членов вложенной структуры.

Вложенные структуры и объединения могут быть вложены без пределов для глубины вложения:

Assembler Выделить код

```
1   struct LINE
2       union
3           start POINT
4           struct
5           x1 dd ?
6           y1 dd ?
7           ends
8       ends
9       union
10          end POINT
11          struct
12          x2 dd ?
13          y2 dd ?
14          ends
15      ends
16  ends
```

Определение структуры может также быть основано на некоторых из уже определенных типов структур, и она наследует всех членов этой структуры, например:

Assembler Выделить код

```
1  struct CPOINT POINT
2      color dd ?
3  ends
```

определяет ту же самую структуру как:

Assembler Выделить код

```
1  struct CPOINT
2      x dd ?
3      y dd ?
4      color dd ?
5  ends
```

Все заголовочные файлы определяют тип данных **CHAR**, который может использоваться, чтобы определить символьные строки в данных структур.

Вернуться к обсуждению:
Мануал по flat assembler

Следующий ответ

0

IT_Exp Эксперт **87844** / 49110 / **22898** Регистрация: 17.06.2006 Сообщений: 92,604	01.09.2014, 06:02

Ошибка в flat assembler
начал изучать ассемблер сталкнулся с такой проблемой: перепечатываю пример из книги: org...

flat assembler массив
У меня есть задание "Упорядочить по убыванию элементы каждого столбцу матрицы" Числа произвольные...

Массив в Flat Assembler
Всем добрый день! Подскажите, почему не работает массив в Flat Assembler? org 100h jmp start...

Как использовать Flat Assembler в Free Pascal?
Я недавно хотел разработать так ради прикола мини ОС с использованием в связке Free Pasal и Flat...

0