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** 99777. **Ответов** 50
Метки (Все метки)

Ответ

2.4 Директивы форматирования

"*format*" со следующим за ним идентификатором формата позволяет выбрать формат вывода. Эта директива должна стоять в начале кода. Формат вывода по умолчанию - это простой двоичный файл, он может быть также выбран директивой "*format binary*".
"*use16*", "*use32*" и "*use64*" указывают ассемблеру генерировать 16-/32-битный или 64-битный код, пренебрегая настройкой по умолчанию для выбранного формата вывода. "use64" включает генерирование кода для длинного режима (*long mode*) процессоров x86.
Ниже описаны разные форматы вывода со специфичными для них директивами.

2.4.1 MZ

Чтобы выбрать формат вывода *MZ*, используйте директиву "*format MZ*". По умолчанию код для этого формата 16-битный.

- "*segment*" определяет новый сегмент, за ним должна следовать метка, чьим значением будет номер определяемого сегмента. Опционально за этой директивой может следовать "*use16*" или "*use32*", чтобы указать битность кода в сегменте. Начало сегмента выровнено по параграфу (16 байт). Все метки, определенные далее, будут иметь значения относительно начала этого сегмента.
- "*entry*" устанавливает точку входа для формата *MZ*, за ней должен следовать дальний адрес (имя сегмента, двоеточие и смещение в сегменте) желаемой точки входа.
- "stack" устанавливает стек для MZ. За директивой может следовать числовое выражение, указывающее размер стека для автоматического создания, либо дальний адрес начального стекового фрейма, если вы хотите установить стек вручную. Если стек не определен, он будет создан с размером по умолчанию в 4096 байт.
- "*heap*" со следующим за ней значением определяет максимальный размер дополнительного места в параграфах (это место в добавление к стеку и для неопределенных данных). Используйте "*heap 0*", Чтобы всегда отводить только память, которая программе действительно нужна.

2.4.2 PE

Чтобы выбрать формат вывода PE, используйте директиву "*format PE*", за ней могут следовать дополнительные настройки формата: используйте "*console*", "*GUI*" или оператор "*native*", чтобы выбрать целевую субсистему (далее может следовать значение с плавающей точкой, указывающее версию субсистемы), "*DLL*" помечает файл вывода как динамическую связывающую библиотеку. Далее может следовать оператор "*at*" и числовое выражение, указывающее базу образа *PE* и далее опционально оператор "*on*" со следующей за ним строкой в кавычках, содержащей имя файла, выбирающей заглушку *MZ* для *PE* программы (если указанный файл не в формате *MZ*, то он трактуется как простой двоичный

исполняемый файл и конвертируется в формат *MZ*). По умолчанию код для этого формата 32-битный. Пример объявления формата *PE* со всеми свойствами:

Assembler Выделить код

```
1      format PE GUI 4.0 DLL at 7000000h on 'stub.exe'
```

"section" определяет новую секцию, за ней должна следовать строка в кавычках, определяющая имя секции, и далее могут следовать один или больше флагов секций. Возможные флаги такие: "code", "data", "readable", "writeable", "executable", "shareable", "discardable", "notpageable". Начало секции выравнивается по странице (4096 байт). Пример объявления секции PE:

Assembler Выделить код

```
1      section '.text' code readable executable
```

Вместе с флагами также может быть определен один из специальных идентификаторов данных PE, отмечающий всю секцию как специальные данные, возможные идентификаторы: "export", "import", "resource" и "fixups". Если секция помечена для содержания настроек адресов, они генерируются автоматически, и никаких данных определять больше не требуется. Также данные ресурсов могут быть сгенерированы автоматически из файлов ресурсов, этого можно добиться, написав после идентификатора "resourse" оператор "from" и имя файла в кавычках. Ниже вы можете увидеть примеры секций, содержащих некоторые специальные данные:

Assembler Выделить код

```
1      section '.reloc' data discardable fixups
2      section '.rsrc' data readable resource from 'my.res'
```

"entry" создает точку входа для PE, далее должно следовать значение точки входа.
"stack" устанавливает размер стека для PE, далее должно следовать значение зарезервиронного размера стека, опционально может следовать отделенное запятой значение начала стека. Если стек не определен, ему присваивается размер по умолчанию, равный 4096 байт.
"heap" выбирает размер дополнительного места для PE, далее должно следовать значение для зарезервированного для него места, опционально ещё может быть значение его начала, отделенное запятой. Если дополнительное место не определено, оно ставится по умолчанию равным 65536 байт, если не указано его начало, то оно устанавливается равным 0.
"data" начинает определение специальных данных PE, за директивой должен следовать один из идентификаторов данных ("export", "import", "resource" или
"fixups") или номер записи данных в заголовке PE. Данные должны быть определены на следующих строках и заканчиваться директивой "end data". Если выбрано определение настроек адресов, они генерируются автоматически, и никаких данных определять больше не требуется. То же самое относится к ресурсам, если за идентификатором "resourse" следует оператор "from" и имя файла в кавычках - в этом случае данные берутся из этого файла ресурсов.

2.4.3 COFF

Чтобы выбрать COFF (Common Object File Format), используйте директиву "format COFF" или "format MS COFF", если вы хотите создать классический мелкософтофский файл COFF. По умолчанию код для этого формата 32-битный. Чтобы создать микросФтовский формат COFF для архитектуры x86-64, используйте установку "format MS64 COFF", в этом случае автоматически будет генерироваться код длинного режима.

- "section" определяет новую секцию, за директивой должна следовать строка в кавычках, определяющая имя новой секции, и ещё может следовать один или более флагов секций. Возможные флаги такие: "code" и "data" для обоих вариантов COFF, "readable", "writeable", "executable", "shareable", "discardable" и "notpageable" только для микросовтофского варианта COFF. По умолчанию секция выровнена по двойному слову (четыре байта), но микросовтовский вариант COFF можно выровнять еще как-нибудь по-другому с помощью оператора "align" и следующим за ним значением выравнивания (любая степень двойки от двух до 8192) среди фагов секций.
- "extrn" определяет внешний символ, за ним должно следовать имя символа и опционально оператор размера, указывающий размер данных, помеченных этим символом. Имя символа также может предваряться строкой в кавычках, содержащей имя внешнего символа и оператор "as". Пара примеров объявления внешних символов:

Assembler Выделить код

```
1       extrn exit
2      extrn '__imp__MessageBoxA@16' as MessageBox:dword
```

- "public" объявляет существующий символ как общедоступный, за ним должно следовать имя символа, и далее опционально оператор "as" и строка в кавычках, содержащая имя, под которым символ будет действителен как общедоступный. Пара примеров объявления общедоступных символов:

Assembler Выделить код

```
1      public main
2      public start as '_start'
```

2.4.4 ELF

Чтобы выбрать формат вывода ELF, используйте директиву "format ELF". По умолчанию код для этого формата 32-битный. Чтобы создать формат ELF для архитектуры x86-64, используйте установку "format ELF", в этом случае автоматически будет генерироваться код длинного режима.

"*section*" определяет новую секцию, за директивой должна следовать строка в кавычках, определяющая имя новой секции, и ещё может следовать один или оба флага "*executable*" и "*writeable*", опционально также может идти оператор "align" со следующим за ним числом, определяющим выравнивание секции (это должна быть степень двойки), если выравнивание не указано, используется значение по умолчанию, которое равно 4 или 8, в зависимости от варианта выбранного формата.
"*extrn*" и "*public*" имеют те же значения и синтаксис у них, как в случае формата COFF (описанного в предыдущем параграфе).
Чтобы создать исполняемый файл, придерживайтесь директивы выбора формата со словом "executable".Это позволяет использовать директиву "entry" со следующим за ним значением, чтобы создать точку входа в программу. С другой стороны это делает недоступными директивы "*extrn*" и "*public*". За директивой "section" в этом случае может следовать только один или более флагов секций и начало секции будет выровнено по странице (4096 байт). Доступные флаги секций такие: "*readable*", "*writable*" и "*executable*".

Вернуться к обсуждению:
Мануал по flat assembler

Следующий ответ

0

Programming
Эксперт
94731 / 64177 / **26122**
Регистрация: 12.04.2006
Сообщений: 116,782

01.09.2014, 06:02

Готовые ответы и решения:

Неофициальная разработка Flat assembler версии 2.0.0
Разработчик Flat assembler'a Tomasz Grysztar в одном из блогов сообщил о разработке новой...

Flat assembler ругается на PROC
Доброго времени суток. Есть программа, собственно вот что она делает: "На экране инициализировать...

Как подключить include к flat компилятору
Здравствуйте,как подключить include к flat компилятору?Требуется подключить include 'win32a.inc' к...

Flat Assembler
Со временем задачи стали нерешаемыми из-за ужасно медленной скорости. Уже давно хочу перейти на...

50