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

# Мануал по flat assembler

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

**Ответ**

## 3.1.8 Ресурсы

Есть два способа создать ресурсы, первый — Вы должны включить внешний файл, сделанного любым компилятором ресурсов например:

Assembler Выделить код

```
1  section '.rsrc' data readable resource from 'my.res'
```

И Вы не должны что-нибудь еще помещать в такую секцию. В случае, если Вы не хотите отдельную секцию для ресурсов, Вы можете поместить их в любую другую секцию с помощью директивы data:

Assembler Выделить код

```
1     data resource from 'my.res'
2     end data
```

Другой способ — Вы должны создать раздел ресурсов вручную. Последний метод, хотя не нуждается ни в какой дополнительной программе, которая будет использована, более трудоемкий, но стандартные заголовочные файлы обеспечивают помощь — набор макросов, которые служат кирпичами, чтобы составить секцию ресурсов.

макроинструкция directory должна быть помещена непосредственно в начале вручную написанных ресурсов, и это определит, какие типы ресурсов используются. Это должно сопровождаться парами значений, первый в каждой паре, является идентификатором типа ресурса, а второй метка подкаталога ресурсов данного типа. Это может выглядеть следующим образом:

Assembler Выделить код

```
1  directory RT_MENU,menus,\
2       RT_ICON,icons,\
3       RT_GROUP_ICON,group_icons
```

Подкаталоги могут быть помещены где-нибудь в секции ресурсов после основного каталога, и они должны быть определены макроинструкцией resource, которая требует, чтобы первый параметр был меткой подкаталога (соответствующий входу в основном каталоге) сопровождаемый тремя параметров — в каждом таком входе, первый параметр определяет идентификатор ресурса (это значение выбранное программистом и используется, чтобы обратиться

к данному ресурсу в программе), второй определяет язык, а третий — метка ресурса. Стандартный equates, должен использоваться, чтобы задавать идентификаторы языка. Например подкаталог меню может быть определен вот так:

Assembler Выделить код

```
1  resource menus,\
2      1,LANG_ENGLISH+SUBLANG_DEFAULT,main_menu,\
3      2,LANG_ENGLISH+SUBLANG_DEFAULT,other_menu
```

Если этот вид ресурсов для которого язык не имеет значения, должен использоваться идентификатор языка LANG_NEUTRAL

Для определения ресурсов различных типов есть специализированные макроинструкции, которые должны быть помещены в секции ресурсов

Точечные рисунки — ресурсы с идентификатором типа RT_BITMAP.

Объявляя растровый ресурс используют макроинструкцию **bitmap** с первым параметром, являющейся меткой ресурса (соответствующей входу в подкаталоге точечных рисунков) и вторым параметром символьной строкой, содержащей путь к bmp-файлу, например:

Assembler Выделить код

```
1  bitmap program_logo,'logo.bmp'
```

Два типа ресурса, связанные со значками — RT_GROUP_ICON — тип ресурса, который должен быть связан с одним или более ресурсам типа RT_ICON, каждый содержащий отдельное изображение. Это позволяет объявлять значки различных размеров и разрядностей цвета под общим идентификатором ресурса. Этот идентификатор, данный ресурсу типа RT_GROUP_ICON можно тогда передать к функции LoadIcon, и она выберет изображение подходящего размера из группы. Чтобы объявлять значок, используйте макроинструкцию icon, с первым параметром, являющимся меткой ресурса RT_GROUP_ICON , сопровождаемого парами параметров, объявляющих изображения. Первым параметром в каждой паре должна быть метка ресурса RT_ICON , и второй символьная строка, содержащая путь к файлу значка. В самом простом варианте, когда группа значков содержит только одно изображение, это будет:

Assembler Выделить код

```
1  icon main_icon,icon_data,'main.ico'
```

где main_icon — метка для входа в подкаталоге ресурса для типа RT_GROUP_ICON, и icon_data — метка для входа типа RT_ICON.

Курсоры объявляют очень похожим способом на значки, на сей раз с типами RT_GROUP_CURSOR и RT_CURSOR и макроинструкцией cursor, которая берет параметры, аналогичные принимаемые макрокомандой icon. Так что определение курсора может выглядеть следующим образом:

Assembler Выделить код

```
1  cursor my_cursor,cursor_data,'my.cur'
```

Меню имеют тип ресурса RT_MENU и объявляется макроинструкцией menu, сопровождаемой несколькими другими макросами определяющими элементы в меню. Само меню берет только один параметр — метка ресурса. menuitem определяет элемент в меню, требуется до пяти параметров, но только два обязательны — первый — символьная строка, содержащая текст пункта меню, и второй — значение его идентификатора (который является значением, которое будет возвращено, когда пользователь выбирает данный элемент в меню). menuseparator определяет разделитель в меню и не требует никаких параметров.

Дополнительный третий параметр menuitem определяет флажки ресурса меню. Есть два таких флажка

- MFR_END — флажок последнего элемента в данном меню, и метках
- MFR_POPUP — флажок что данный элемент — подменю, и следующие элементы будут элементами, составляющими то подменю, пока элемент с флажком MFR_END не найден.

Флажок MFR_END можно также передать как параметр для menuseparator и это единственный параметр, который эта макрокоманда может взять. Каждое подменю должно быть закрыто элементом с флажком MFR_END, и полностью меню должно также быть закрыто этим флагом. Вот — пример законченного объявления меню:

Assembler Выделить код

```
1  menu main_menu
2      menuitem '&File',100,MFR_POPUP
3      menuitem '&New',101
4      menuseparator
5      menuitem 'E&xit',109,MFR_END
6  menuitem '&Help',900,MFR_POPUP + MFR_END
7      menuitem '&About...',901,MFR_END
```

Дополнительный четвертый параметр menuitem определяет флажки состояния за данного элемента, эти флажки те же самые как те что используются функциями API, подобно MFS_CHECKED или MFS_DISABLED. Точно так же пятый параметр может определить типа флажка. Например, это определит элемент отмеченный с отметкой переключателя:

Assembler Выделить код

```
1  menuitem ’Selection’,102, ,MFS_CHECKED,MFT_RADIOCHECK
```

Диалоговые окна имеют тип ресурса RT_DIALOG и объявляются макроинструкцией **dialog**, сопровождаемой любым количеством элементов, начатых dialogitem и законченных enddialog.

**dialog** может взять до одиннадцати параметров, минимум требуется семь.

- Первый параметр, как обычно, определяет метку ресурса,
- второй — символьная строка, содержащая заголовок диалогового окна,
- следующие четыре параметра определяют горизонтальные и вертикальные координаты, ширину и высоту диалогового окна соответственно.
- Седьмой параметр определяет флажки стиля для диалогового окна,
- дополнительный восьмой определяет расширенные флажки стиля.
- Девятый параметр может определить меню для окна — это должен быть идентификатор ресурса меню, такой же самый как один из указанных в подкаталоге ресурсов с типом RT_MENU.
- Наконец десятый и одиннадцатый параметр может использоваться, чтобы определить шрифт для диалогового окна — сначала , должна быть символьная строка, содержащая название шрифта, и после нее номер, определяющий размер шрифта. Когда эти дополнительные параметры не определены, используется значение по умолчанию, MS Sans Serif размером 8 кеглей.

Этот пример показывает макрокоманду dialog со всеми параметрами кроме меню (который оставляют с пустым значением), дополнительные параметры находятся во второй строке:

Assembler Выделить код

```
1  dialog about,’About’,50,50,200,100,WS_CAPTION+WS_SYSMENU,\
2      WS_EX_TOPMOST, ,’Times New Roman’,10
```

dialogitem имеет восемь обязательных параметров и один дополнительный

- Первым параметром должна быть символьная строка, содержащая имя класса элемента управления.
- Вторым параметром может быть или символьная строка содержащий текст на элементе, или идентификатор ресурса в случае, если, когда содержание элемента должно быть определено некоторым дополнительным ресурсом (подобно элементу класса STATIC с стилем SS_BITMAP).
- Третий параметр — идентификатор элемента, который идентифицирует элемент для функций API.
- Затем четыре параметра определяют горизонтальные, вертикальные координаты, ширину и высоту элемента соответственно.
- Восьмой параметр определяет, стиль элемента,
- и дополнительный девятый определяет расширенный стиль.

Пример определения элемента диалога:

Assembler Выделить код

```
1  dialogitem 'BUTTON','OK',IDOK,10,10,45,15,WS_VISIBLE+WS_TABSTOP
```

И пример статического элемента, содержащего точечный рисунок, предполагая, что там существует растровый ресурс с идентификатором 7:

Assembler Выделить код

```
1  dialogitem ’STATIC’,7,0,10,50,50,20,WS_VISIBLE+SS_BITMAP
```

Определение ресурса диалога может содержать любое количество элементов или ни одного вообще, и должно всегда заканчиваться макроинструкцией enddialog.

Ресурсы типа RT_ACCELERATOR создаются с макроинструкцией **accelerator**. Первый параметр, традиционно является меткой ресурса, далее должны следовать за три параметра — флажки акселератора, сопровождаемые кодом виртуальной клавиши или символом ASCII и значением идентификатора (который подобен идентификатору пункта меню). Простое определение акселератора может выглядеть следующим образом:

Assembler Выделить код

```
1  accelerator main_keys,\
2      FVIRTKEY+FNOINVERT,VK_F1,901,\
3      FVIRTKEY+FNOINVERT,VK_F10,109
```

Информация версии — ресурс типа RT_VERSION и создается макросом version. После метки ресурса, второй параметр определяет операционную систему PE файла (обычно VOS __ WINDOWS32), третий параметр, тип файла (наиболее общий является VFT_APP для программы и VFT_DLL для библиотеки), четвертый подтип (обычно VFT2_UNKNOWN), пятый идентификатор языка, шестой кодовая страница и затем пары из символьных строк — названия свойств и определяющие их значения. Самая простая информация версии может быть определена вот так:

Assembler Выделить код

Assembler Выделить код

```
1  versioninfo vinfo,VOS__WINDOWS32,VFT_APP,VFT2_UNKNOWN,\
2      LANG_ENGLISH+SUBLANG_DEFAULT,0,\
3      'FileDescription','Description of program',\
4      'LegalCopyright','Copyright et cetera',\
5      'FileVersion','1.0',\
6      'ProductVersion','1.0'
```

Другие виды ресурсов могут быть определены макроинструкцией **resdata**, которая берет только один параметр — метка ресурса, и может сопровождаться любыми командами, определяющими данные,а заканчиваться макроинструкцией endres, например:

Assembler Выделить код

```
1  resdata manifest
2      file 'manifest.xml'
3  endres
```

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, 06:06

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**