Proteus90 8 мая 2015 в 13:30

Форматы файлов для программ на FASM под Windows

Assembler

Из песочницы

При создании программы на ассемблере (для примера будет приведён FASM) из-под ОС Windows возникает вопрос о том, какой выбрать формат файла.
Для определения формата создаваемого исполняемого файла используется директива «format» со следующим за ним идентификатором формата.
Под катом краткое описание COM-программы и EXE-программы форматов MZ и PE с шаблоном программ (в виде традиционного «Hello World!»).

Формат по умолчанию — это простой двоичный файл, он также может быть выбран директивой «format binary», формирующий программы типа .COM.
«use16» и «use32» указывают ассемблеру генерировать 16-битный или 32-битный код, пренебрегая настройкой по умолчанию для выбранного формата вывода. «use64» включает генерирование кода для длинного режима процессоров x86.
Ниже описаны разные форматы вывода со специфичными для них директивами.

.COM-программы

Программы типа .com после загрузки в память представляют собой немодифицированное представление программы на машинном языке на диске. Формат .com является одним из простейших форматов исполняемых файлов на архитектуре x86. Размер .com-файла ограничен размером 1 сегмента и равен 64 КБ, все данные должны быть определены в этом же сегменте кода. Когда COM-программа начинает работать, все сегментные регистры содержат адрес префикса программного сегмента (PSP), — 256-байтового (100h) блока, который резервируется операционной системой DOS непосредственно перед COM или EXE программой в памяти. Так как адресация начинается со смещения 100h от начала PSP, то в программе кодируется директива ORG 100h. Директива эта устанавливает относительный адрес для начала выполнения программы. Программный загрузчик использует этот адрес для командного указателя.

Пример простой программы в формате .COM:

```
use16               ;Генерировать 16-битный код
org 100h            ;Программа начинается с адреса 100h

    mov dx,hello    ;В DX адрес строки.
    mov ah,9        ;Номер функции DOS.
    int 21h         ;Обращение к функции DOS.

    mov ax,4C00h    ;В регистр AH помещаем 4Ch, в AL - 00h.
    int 21h         ;Завершение программы
;--------------------------------------------------------
hello db 'Hello, world!$'
```

Директива «use 16» указывает на генерирование 16-битного кода. «org 100h» объявляет пропуск 256 байт (адреса 0000h – 00FFh). Указанные адреса зарезервированы под служебные данные (PSP).
Далее следуют команды. В регистр DX помещается адрес строки hello. Затем вызывается функция номер 9 прерывания 21h для вывода строки на экран.
Завершение работы программы осуществляется вызовом функции 4C с параметром того же прерывания 21h.
Строка hello завершается символом '$', который в системе DOS обозначает конец строки.

Следует помнить, что программы типа COM не поддерживаются 64-разрядными ОС Windows. Для запуска таких программ под

этими операционными системами следует использовать программу DOSBox, либо воспользоваться форматом PE, рассмотренном ниже.

Формат MZ

MZ — стандартный формат 16-битных исполнимых файлов с расширением .EXE для DOS. Назван так по сигнатуре — ASCII-символам MZ (4D 5A) в первых двух байтах.

Пример простой программы с использованием формата MZ:

```
format MZ                       ;Исполняемый файл DOS EXE (MZ EXE)
entry code_seg:start            ;Точка входа в программу
stack 200h                      ;Размер стека
;-------------------------------------------------------------------
segment data_seg                ;Сегмент данных
    hello db 'Hello, asmworld!$'    ;Строка
;-------------------------------------------------------------------
segment code_seg                ;Сегмент кода
start:                          ;Точка входа в программу
    mov ax,data_seg             ;Инициализация регистра DS
    mov ds,ax

    mov ah,09h
    mov dx,hello                ;Вывод строки
    int 21h

    mov ax,4C00h
    int 21h                     ;Завершение программы
```

Для создания нужно использовать 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 разрядность кода в сегменте. Начало сегмента выровнено по параграфу (16 байт). Все метки, определенные далее, будут иметь значения относительно начала этого сегмента. В примере выше объявляются 2 сегмента: «data_seg» и «code_seg».
«entry» устанавливает точку входа для формата MZ, за ней должен следовать дальний адрес (имя сегмента, двоеточие и смещение в сегменте) желаемой точки входа. В нашем случае объявлена метка «start».
«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.

Формат MZ, аналогично COM-программам, не поддерживается 64-рязрядными ОС Windows.

Формат PE

PE — это сокращение от Portable Executable, т.е. переносимый (универсальный) исполняемый файл. Этот формат появился еще в поздние времена Windows 3.11, но настоящее распространение получил с расцветом Windows 95. Можно сказать, что сейчас на компьютерах с Windows 9x/2K/XP/Vista/7 находится 95% исполняемых (exe, dll, драйверы(sys)) файлов — это PE файлы.

Чтобы выбрать формат PE, нужно использовать директиву «format PE», за ней могут следовать дополнительные настройки формата: «console», «GUI» или оператор «native», чтобы выбрать целевую подсистему (далее может следовать значение с плавающей точкой, указывающее версию подсистемы), «DLL» помечает файл вывода как динамическую связывающую библиотеку. Далее может следовать оператор «at» и числовое выражение, указывающее базу образа PE, и 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:
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:
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:
section '.reloc' data discardable fixups
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 зарезервированного 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.

Пример простой программы с использованием формата PE:

```
format PE console              ;Исполняемый файл Windows EXE
entry start                    ;Точка входа в программу
include 'win32a.inc'
section '.text' code executable
start:
    push hello
    call [printf]
    push 0
    ccall   [getchar]
    call [ExitProcess]
section '.rdata' data readable
    hello db 'Hello World!', 0
section '.idata' data readable import
    library kernel32, 'kernel32.dll', \
```


В данном примере для работы с консолью использованы функции WinAPI.

Вот такой получился краткий (надеюсь, для кого-то полезный) обзор использования форматов PE и MZ. За бортом данной статьи оказались ELF и COFF, за что прошу сильно не судить.

Теги: FASM, assembler, форматы файлов

Хабы: Assembler

Комментарии 3

CoolWolf 8 мая 2015 в 13:38 +10

Мне кажется, что это знает каждый, кто знаком с ассемблером...

pravic 9 мая 2015 в 17:09 0

Перевод четырёх абзацев мануала, зачем?

Sergey6661313 20 февраля 2018 в 10:28 0

Какого мануала? Киньте ссылку. И вообще почему нет? переводы всегда хорошо...

САМОЕ ЧИТАЕМОЕ

Сутки | Неделя | Месяц

Как потерять аккаунт на Гос. услугах за 5 секунд
+105 30,5k 52 142

Эпидемия летаргии: заснувшие навсегда
+132 43,2k 87 113

Проверь свои клеммники
+57 27,4k 58 196

Домашний тлен: токсины черной плесени
+74 17,7k 91 71

Карты, деньги, Data Science: изучаем нескучные банковские данные [КВЕСТ]
Мегапост

Ваш аккаунт
Войти
Регистрация

Разделы
Публикации
Новости
Хабы

Информация
Устройство сайта
Для авторов
Для компаний

Услуги
Реклама
Тарифы
Контент

Хабы Для компаний Контент

Компании Документы Семинары

Пользователи Соглашение Мегапроекты

Песочница Конфиденциальность Мерч

Настройка языка **О сайте** **Служба поддержки** **Мобильная версия**