ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
ФИЗТЕХ-ШКОЛА РАДИОТЕХНИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Лабораторная работа по программированию.

По теме

«Исследование структуры exe файла»

Студента 1 курса группы Б01-003
Крейнина Матвея Вадимовича

Долгопрудный, 2021

Исследование структуры exe файла

Цель работы:

Изучить структуру работы exe файла и транслировать свой ассемблер в рабочий exe файл.

В работе используются:

Кривые руки Матвея Крейнина, библиотека mk5.dll , написанная им же в одну из прекрасных майских ночей, Visual Studio с её красивым и невообразимым дебагом, подсветкой, окном любых переменных, ходьбой по шагам и мягким seek fault-ом. К сожалению, в этой работе это отсутствовало 90% времени, но было найдено решение ImpEx для far manager, которое позволило меньше взаимодействовать с hex-ами.

Теоретическое введение:

Для начала я хочу начать с того, что нужно написать свои упрощенные функции scanf и printf, потому что стандартные работают со сторокой, что не очень удобно, также нужно было реализовать функцию завершения программы, её я нашел в библиотеке kernel32.dll. Далее мне помогла следующая статья, How to build a DLL in MASM32. Нужно особым образом указывать вход в саму библиотеку и дальше обрабатывать аргументы, поскольку я использую dll только для подключения я по стандарту «пихаю правду» в регистр eax.

```
;=====================================================================
;       Entry point for my library
;=====================================================================
LibMain proc parameter1:DWORD, parameter2:DWORD, parameter3:DWORD

        mov eax, 1
        ret

LibMain endp
;=====================================================================
```

Всё остальное также, как и при написании в обычном asm файле, ничего сложного. Вот пример компиляции самой библиотеки.

```
@echo off

if exist mk5.obj del mk5.obj
if exist mk5.dll del mk5.dll

C:\masm32\bin\ml /c /coff mk5.asm

C:\masm32\bin\Link /SUBSYSTEM:WINDOWS /DLL /DEF:mk5.def mk5.obj

dir mk5.*
```

Для компиляции нужно составить def файл, в нём вы укажите имя библиотеки и названий функций, которые вы хотите экспортировать, сделать это нужно, вот так:

```
LIBRARY mk5lib
EXPORTS WriteConsoleA
EXPORTS ReadConsoleA
EXPORTS ExitProcess
EXPORTS ToDec
EXPORTS mk5scanf
```

Поздравляю, теперь вам стоит отдебажить это, и проверить не зануляют ли ваши функции регистры, а то мало ли что может из этого выйти.

Ход работы:

1. Сделать отдельную функцию translate, которая отвечает за трансляцию своего ассемблера в nasm, для этого была сделана вспомогательная функция заполнения таблицы переходов. При первом проходе кода она заполняет массив переходов, запоминая откуда и куда. Далее при втором проходе проверяется, что это адрес назначения для какого-нибудь перехода или нет, в отдельное поле уже заносится адрес перехода уже в x86, таким образом решена проблема с переходами. Для удобства и дальнейшего расширения проекта был добавлен DSL, так с его помощью можно добавлять команды, как char-ы, так и word-ы, адреса переходов и числа. Благодаря этому увеличилась скорость написания дополнительных команд.

2. Но теперь перейдём к самому интересному - exe файлу. Так получилось, что компания microsoft хотела обратной совместимости с dos и для этого придумала dos-заглушку, которая абсолютно стандартна для всех программ. В досе она выводит, что данная программа не может быть запущена. Она стандартная и я ёе нашел в интернете и вставил в свой проект. Но Марк Збиковский красиво внёс себя в историю на многие десетелетия. Далее идёт file header, рассмторим в нём некоторые поля. Number of sections, которое отвечает за количество секций в вашем exe файле, в моём случае это .code, .idata и .data. Хочу обратить ваше внимание на поле Machine(014C), рекомендую вам оставить его таким и не менять его под ваш ноутбук, особенно, если вы пишете в первый раз свой exe-шник, в противном случае, можно потерять много времени, пока вы не найдете эту ошибку. Также нужно будет указать стандартный размер optional header-a.

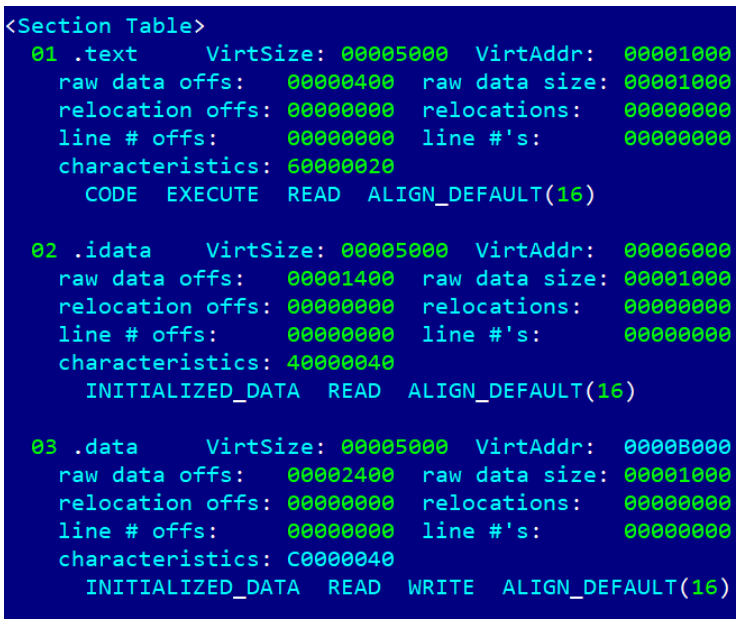
```
<File Header>
  Machine:                      014C (I386)
  Number of Sections:           0003
  TimeDateStamp:                6099576D -> Mon May 10 18:55:25 2021
  PointerToSymbolTable:         00000000
  NumberOfSymbols:              00000000
  SizeOfOptionalHeader:         00E0
  Characteristics:              0102
    EXECUTABLE_IMAGE
    32BIT_MACHINE
```

3. Перейдем к optional header-у, в нём нужно указать магическое число 10, которое отвечает за 32-х битную систему. Потому нужно указать точку входа в программу,

по стандарту, это 0x1000, далее нужно указать размер каждой секции, размер image base, выравнивание заголовков, размер заголовгов и subsystem и количество дата директорий (rvas and sizes).

```
<Optional Header>
  Magic                         010B
  linker version                0.00
  size of code                  0
  size of initialized data      0
  size of uninitialized data    0
  entrypoint RVA                1000
  base of code                  0
  base of data                  0
  image base                    400000
  section align                 1000
  file align                    200
  required OS version           0.00
  image version                 0.00
  subsystem version             4.00
  Win32 Version                 0
  size of image                 10000
  size of headers               400
  checksum                      0
  Subsystem                     0003 (Windows character)
  DLL flags                     0000
  stack reserve size            0
  stack commit size             0
  heap reserve size             0
  heap commit size              0
  RVAs & sizes                  10
```

4. Предлагаю узнать об очень важной области exe файла - sectional table. В нём нужно указывать виртуальный размер секции, также нужно указать адрес в файле, где лежит эта секция, размер этой секции в файле и виртуальный адрес, по нему будет в дальнейшем будет адресоваться в самой программе, когда она подгрузится в память. Тут всё просто, 400 столько ушло на заголовки, виртуальный 1000, столько мы указали в optional header-е в поле entrypoint RVA и ставим параметры. Для .idat-ы и .data, делаем всё тоже самое. Важно заметить, что если у вас оказалось меньше данных, чем вы указали в raw-data-size, то это место нужно заполнить нулями, так чтобы адрес начала других секций не сместился. Поздравляю, если вы указали все поля правильно, а также решили не выпендриваться в file header-е в поле machine, то у вас будет должна запуститься программа.



Вывод

Время подсчёта факториала 10 один миллиард раз в exe файле составило 19.25 секунд. Время подсчёта факториала 10 тысячу раз на моём процессоре составило 0.29 секунд. Проведя несложную арифметику я сделал вывод, что подсчёт одного факториала стал в 15000 раз быстрее, чему я очень рад, наверное, и не только я. Я понял, почему так важно указывать источники при составлении статей, прилагаю все, которые были полезны.

Оптимизации

Было решено сделать оптимизации. Сначала я решил оптимизировать машинные коды процессора, потом я понял, что я не такой умный, как процессор и не могу понять какой длины у меня команды, поэтому было решено в промежуточный формат добавить дополнительные команды, которых не было в моём ассемблере. Это примеры упрощений, которые были выбраны оптимизировать. Первые два это присваивания регистра или числа в другой регистр, следующие 3 упрощения связаны с добавлением или вычитанием числа из регистра, это упрощение было выбрано для того, чтобы у меня ускорилась работа всех циклов.

```
//Примеры для упрощения:
/*
1.  push 1
    pop reg
    - mov reg, 1

2.  push reg1
    pop reg2
    - mov reg2, reg1

3.  push num
    push reg1
    add
    pop reg1
    - add reg1, num

4.  push reg1
    push num
    add
    pop reg1
    - add reg1, num

5. push reg1
   push num
   sub
   pop reg1
   - sub reg1, num
*/
```

Вывод по оптимизациям

После оптимизации получилось 11.54 секунд при подсчёте факториала 10 один миллиард раз, т.е. ускорение по сравнению с прошлым exe получилось 1.66 раз. Но было решено изменить способ измерения времени на моём процессоре, и я изменил режим компиляции exe-файла для процессора и получил 0,84 секунды на процессоре при подсчётн факториала 10 сто тысяч раз.При изменении режима дебага на релиз, мой процессор стал работать быстрее, но ускорение всё равно довольно заметно, а именно в 728 раз, что я еще считаю очень хорошим результатом.

Список литературы

1. Создаём exe файл https://habr.com/ru/post/515058/

   Эта статья даст вам более широкое понимание того, как нужно заполнять заголовки в exe файле.

2. PE (Portable Executable): На странных берегах https://habr.com/ru/post/266831/

   Эта статья заходит с более «журналисткой» точки зрения того, что происходит в exe файле без указания конкретных параметров.

3. PE-файлы: общее описание

   http://cs.usu.edu.ru/docs/pe/

   Вот тут уже техническое описание каждого раздела, с указаными структурами из библиотек для C и C++.

4. Пошаговое руководство к исполняемым файлам (EXE) Windows

   https://habr.com/ru/post/148194/

   Красивая картинка, которая поможет структурировать знания об устройстве exe файла.

5. Какая-то немецкая статья на английском, которая описывает структуру exe файла.

   http://www.zotteljedi.de/pub/pe.txt

6. Плагин для красивого вида exe файла, который поможет в дальнейшем дебаге, если вы конечно не любитель копаться в чистых hexах.

   https://plugring.farmanager.com/plugin.php?l=ru & pid=790

7. How to build a DLL in MASM32

   Гайд по созданию своей динамической библиотеки.

   http://www.website.masmforum.com/tutorials/dlltute/masmdll.htm

8. Дединский Илья Рудольфович http://ded32.net.ru

   Возможно, вы заходите заняться компутерной графикой и тогда вы сможете найти одну библиотеку, которая сможет вам помочь.

9. Google https://www.google.ru