МИНОБРНАУКИ РФ

ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

Отчет

по лабораторной работе № 4 по дисциплине

на тему "Анализ структуры объектных и загрузочных файлов"

"Машинно-зависимые языки программирования"

Вариант 5

Выполнил студент группы Б04-191-3: Р.А. Гумметов

Принял: В.Г. Тарасов

Ижевск 2020

1. Постановка задачи

Ход выполнения стандартной лабораторной работы:

1. Взять тестовый пример из лабораторной работы 3, создать исходный файл, например, test.asm.
2. С помощью транслятора tasm.exe получить листинг для примера test.lst. Также создается объектный файл test.obj.
3. Распечатать dump листинга в 16-м виде, разобрать из каких записей он состоит, дать объяснение по содержимому записей.
4. С помощью компоновщика tlink.exe из test.obj получить загрузочный файл test.exe для подготовленного примера.
5. Распечатать dump загрузочного файла test.exe в 16-м виде, разобрать его по отдельным составляющим, дать объяснение этих составляющих.

Исходный файл, на основе 3 лабораторной работы:

Файл test.asm

dseg segment

dseg ends

cseg segment

assume cs:cseg, ds:dseg

start far

mov ax, dseg

mov ds, ax

metka2:

sub ax, bx

sub cx, di

metka3:

sub al, bl

sub cx, [BX]

sub di, [SI]

sub dl, [SI]

sub bx, [DI]

metka1:

sub bx, [BX + DI]

cmp ax, bx

jne metka1

cmp cx, [BX]

je metka2

cmp bx, [BX + DI]

je metka3

cmp bl, [BX + SI]

mov ax, 4C00h

int 21h

cseg ends

endstart

2. Ассемблер

Ассемблер - транслятор исходного текста программы, написанной на языке ассемблера, в программу на машинном языке.

Ассемблер получает файл .objиз исходного ассемблерного кода .asm. Для того, чтобы получить файл .obj, выполним команду:

tasm.exetest.asm /l

/lнужно чтобы также получить листинг нашей программы.

Листинг test.lst

1 0000 dseg segment

2

3 0000 dseg ends

4 0000 cseg segment

5 assume cs:cseg, ds:dseg

6 0000 start:

7 0000 B8 0000s mov ax, dseg

8 0003 8E D8 mov ds, ax

9

10 0005 metka2:

11 0005 2B C3 sub ax, bx

12 0007 2B CF sub cx, di

13 0009 metka3:

14 0009 2A C3 sub al, bl

15 000B 2B 0F sub cx, [BX]

16 000D 2B 3C sub di, [SI]

17 000F 2A 14 sub dl, [SI]

18 0011 2B 1D sub bx, [DI]

19 0013 metka1:

20 0013 2B 19 sub bx, [BX + DI]

21 0015 3B C3 cmp ax, bx

22 0017 75 FA jne metka1

23 0019 3B 0F cmp cx, [BX]

24 001B 74 E8 je metka2

25 001D 3B 19 cmp bx, [BX + DI]

26 001F 74 E8 je metka3

27 0021 3A 18 cmp bl, [BX + SI]

28

29 0023 B8 4C00 mov ax, 4C00h

30 0026 CD 21 int 21h

31

32 0028 csegends

33 endstart

В листинге расположена таблица. В первой колонке находятся номера строк исходного файла test.asm. Во второй колонке текущий адрес команда в hex. Далее перевод текущей команда ассемблера в машинный код и сама команда.

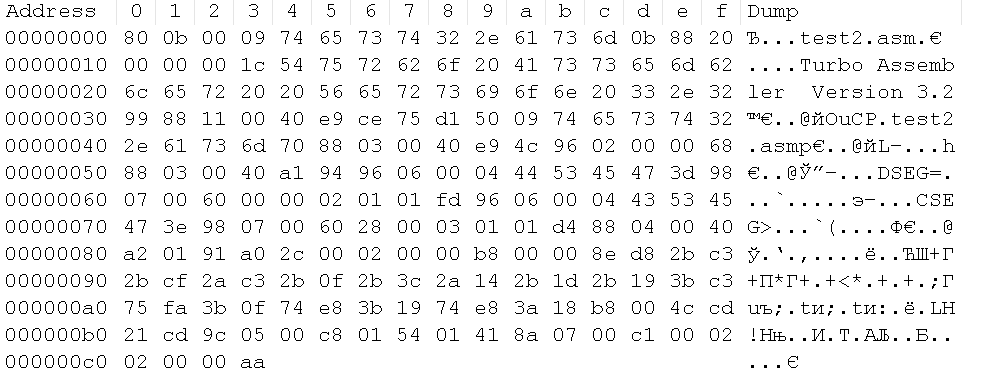
Важно, что около объектных кодов нескольких команд стоят буквы r, s, se. Таким образом Ассемблер сообщает о проблемах, с которыми он не может справиться, поскольку не обладает данными о расположении программных объектов. Далее мы увидим, что с этими отметками связаны записи FIXUPP в obj-файлах.

Отметим, что r-проблемы – это «историческое» наследие, соответствующие записи FIXUPP хоть и строятся, но не обрабатываются компоновщиком. А s-проблема, связанная положением в памяти сегмента данных, будет решаться компоновщиком, а еще и загрузчиком.

Последний существенный момент – директива endstart, в которой обозначена точка входа, что необходимо в файле с главной программой.

Рассмотрение листингов, являющихся вспомогательным выводом Ассемблера, позволяет понять содержательный смысл примера.

Dump объектного файла test.obj

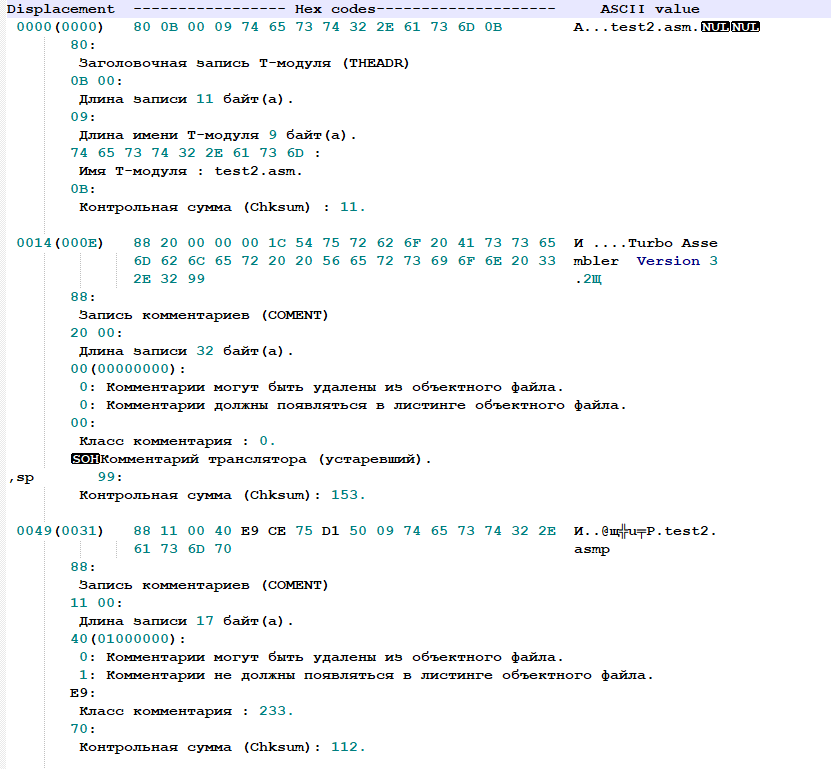


Проанализируем объектный файл с помощью objview.exe. Для этого введем команду.

objview.exetest.obj

В файле test.OUT теперь находится полный разбора объектного файла.

Начало файла test.OUT



Краткий разбор объектного файла:

80 0B 00 09 74 65 73 74 32 2E 61 73 6D 0B - Заголовочная запись T-модуля (THEADR)

88 20 00 00 00 1C 54 75 72 62 6F 20 41 73 73 656D 62 6C 65 72 20 20 56 65 72 73 69 6F 6E 20 332E 32 99- Запись комментариев (COMENT)

88 11 00 40 E9 CE 75 D1 50 09 74 65 73 74 32 2E61 73 6D 70- Запись комментариев (COMENT)

88 03 00 40 E9 4C - Запись комментариев (COMENT)

96 02 00 00 68- Запись списка имен (LNAMES)

88 03 00 40 A1 94- Запись комментариев (COMENT)

96 06 00 04 44 53 45 47 3D- Запись списка имен (LNAMES)

98 07 00 60 00 00 02 01 01 FD - Запись определения сегмента (SEGDEF)

96 06 00 04 43 53 45 47 3E- Запись списка имен (LNAMES)

98 07 00 60 28 00 03 01 01 D4- Запись определения сегмента (SEGDEF)

88 04 00 40 A2 01 91 - Запись комментариев (COMENT)

A0 2C 00 02 00 00 B8 00 00 8ED8 2BC3 2BCF 2AC3 2B 0F 2B 3C 2A 14 2B 1D 2B 19 3B C3 75 FA 3B0F 74 E8 3B 19 74 E8 3A 18 B8 00 4C CD 21 CD-Логическая упорядоченная запись данных (LEDATA)

9C 05 00 C8 01 54 01 41- Запись привязки (FIXUPP)

8A 07 00 C1 00 02 02 00 00 AA- Конечная запись модуля (MODEND)

Для изучения алгоритмов компоновщика нужны структуры obj-файлов, которые были получены с помощью утилиты tdump вызываемой так:

tdumptest.objtest.oud

Файл test.oud:

Turbo Dump Version 3.1 Copyright (c) 1988, 1992 Borland International

Display of File TEST2.OBJ

000000 THEADR test2.asm

00000E COMENT Purge: Yes, List: Yes, Class: 0 (000h)

Translator: Turbo Assembler Version 3.2

000031 COMENT Purge: Yes, List: No , Class: 233 (0E9h)

Dependency File: test2.asm 06/17/120 02:46 pm

000045 COMENT Purge: Yes, List: No , Class: 233 (0E9h)

End of Dependency List

00004B LNAMES

Name 1: ''

000050 COMENT Purge: Yes, List: No , Class: 161 (0A1h)

Object Module Extensions Present.

000056 LNAMES

Name 2: 'DSEG'

00005F SEGDEF 1 : DSEG PARA PRIVATE Length: 0000

000069 LNAMES

Name 3: 'CSEG'

000072 SEGDEF 2 : CSEG PARA PRIVATE Length: 0028

00007C COMENT Purge: Yes, List: No , Class: 162 (0A2h)

Linker - Pass Two Marker.

000083 LEDATA Segment: CSEG Offset: 0000 Length: 0028

0000: B8 00 00 8E D8 2B C3 2B CF 2A C3 2B 0F 2B 3C 2A .....+.+.\*.+.+<\*

0010: 14 2B 1D 2B 19 3B C3 75 FA 3B 0F 74 E8 3B 19 74 .+.+.;.u.;.t.;.t

0020: E8 3A 18 B8 00 4C CD 21 .:...L.!

0000B2 FIXUPP

FixUp: 001 Mode: Seg Loc: Base Frame: TARGET Target: SI[1]

0000BA MODEND(Main Module) Frame: SI[2] Target: SI[2], 0000h

При работе компоновщика могут появляться различные проблемы, *r*, *s, se*упомянутые выше.

FIXUPP’ы типа Offset16 соответствуют описанным r-проблемам и компоновщиком не изменяют значения смещений, поскольку аргументом в методе определения цели Target указаны поля сегментов.

FIXUPP типа Pointer16 соответствует se-проблеме, его поля обозначают следующее.

FIXUPP типа Base соответствует s-проблеме, его поля обозначают следующее.

FixUp: 001 – начиная со смещения 001 в предыдущей записи LEDATA есть 2 байта (об этом говорит тип Base), в которые транслятор не смог положить достоверную информацию;

Mode: Seg – говорит о межсегментном типе привязки;

Frame: TARGET – говорит о том, что метод определения фрагмента определяется методом определения цели Target;

Target: SI[2] – говорит, что метод определения цели зависит от размещения в памяти сегмента 2 (SI[2]).

Важно, что записи LEDATA построены для сегментов DSEG и CSEG, но отсутствуют для SSEG, поскольку содержимое стекового сегмента при запуске не имеет значения.

3. Структура заголовка .exe файла



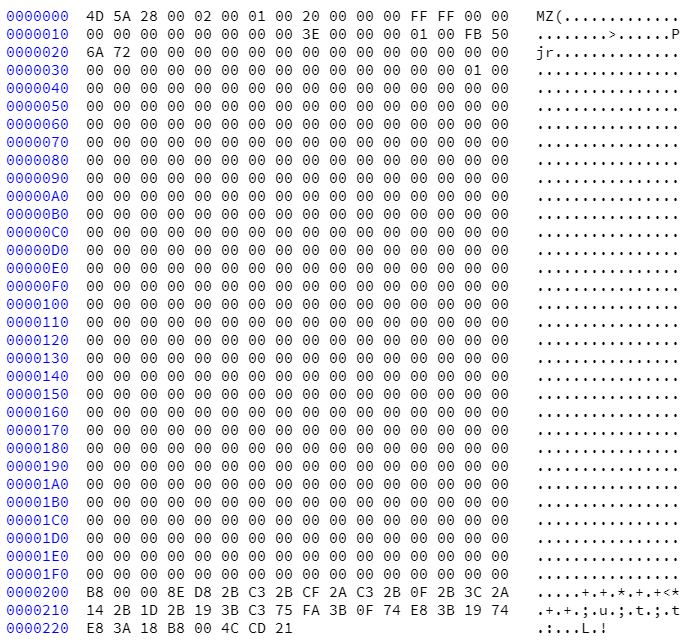
4. Загрузочный файл

Назначение системной программы загрузчика – перенос тела выполняемой программы из загрузочного (выполняемого) файла в оперативную память, настройка под выделенное адресное пространство адресных полей, зависящих от расположения в основной памяти.

Загрузчик руководствуется информацией от компоновщика, сформированной в заголовке. ехе-файла. Загрузчик, зная размер выполняемой программы, запрашивает у операционной системы необходимый непрерывный участок оперативной памяти. Переписывает в эту память тело программы из .ехе-файла. Пользуясь таблицей перемещения, производит настройку под выделенное адресное пространство адресных полей, зависящих от расположения в основной памяти. Затем инициализируются регистры SS, SP, CS, IP и управление передается выполняемой программе

Для того, чтобы получить .exeфайл из объектного файла .objнужно выполнить команду:

tlinktest.obj

Dump загрузочного файла test.exe

Заголовочная часть:

Значение байт:

4D 5A “Магическое число”  
2800 Количество байтов последней страницы программы  
02 00 Размер exe файла в страницах по 512 байт  
01 00 Количество элементов в таблице перемещения  
20 00 Количество параграфов в заголовке файла  
00 00 Минимальное количество параграфов памяти  
FFFF Максимальное количество параграфов дополнительнойпамяти  
00 00 Смещение сегмента стека от начала тела программы  
00 00 Начальное значение SP  
00 00 Контрольная сумма (без ошибок - 0)  
00 00 Начальное значение IP  
00 00 Смещение в параграфах от начала тела программы для кодового сегмента  
3E 00 Смещение таблицы перемещения в exe  
00 00 Номер оверлея (0 – главная программа)

Сегментная часть:

Из Dump файла объектного файла можно узнать размеры всех сегментов: DSEG – 0 байт, CSEG - 40 байта. Отсчитывая от конца 40 байт – это будет segmentCSEG и в начале заголовок exe-файла. Как видно все значения в CSEG совпадают с листингом программы на ассемблере.

Для увеличения оперативности анализа можно получить более наглядную информацию из заголовка если применить следующую команду

tdump –e test.exe test.hid

Файлtest.hid

Turbo Dump Version 3.1 Copyright (c) 1988, 1992 Borland International

Display of File TEST2.EXE

DOS File Size 237h ( 567. )

Load Image Size 37h ( 55. )

Relocation Table entry count 0001h ( 1. )

Relocation Table address 003Eh ( 62. )

Size of header record (in paragraphs) 0020h ( 32. )

Minimum Memory Requirement (in paragraphs) 0000h ( 0. )

Maximum Memory Requirement (in paragraphs) FFFFh ( 65535. )

File load checksum 0000h ( 0. )

Overlay Number 0000h ( 0. )

Borland TLINK Version 5.00

Initial Stack Segment (SS:SP) 0000:0000

Program Entry Point (CS:IP) 0000:0000

RelocationLocations (1 Entry)

0000:0001

Наиболее существенно, что после загрузки в основную память должны будут измениться на величину адреса загрузки одно адресное поле, на что указывает элемент таблицы перемещения. Вспомним, что это были сегментная часть адреса сегмента данных (величина загружается в регистр DS). Чтобы увидеть эти изменения, воспользуемся еще одним инструментом – резидентной программой mydump, которая «прослушивает» прерывание с номером 70h. Программа mydump запускается перед анализируемой программой. Анализируемая программа фиксирует адрес оперативной памяти из своего тела и количество байтов, содержимое которых с помощью mydump записывается в файл; имя файла вводится с клавиатуры. Сама же mydump вызывается через прерывание 70h, вспомним, что в программе main2020.asm были строки, ответственные за подготовку параметров и вызов mydump.

Для этого нужно немного изменить программу.

Измененная программа test2.asm:

dseg segment

dseg ends

cseg segment

assume cs:cseg, ds:dseg

start proc far

mov ax, dseg

mov ds, ax

metka2:

sub ax, bx

sub cx, di

metka3:

sub al, bl

sub cx, [BX]

sub di, [SI]

sub dl, [SI]

sub bx, [DI]

metka1:

sub bx, [BX + DI]

cmp ax, bx

jne metka1

cmp cx, [BX]

je metka2

cmp bx, [BX + DI]

je metka3

cmp bl, [BX + SI]

;===============================================

lea cx,eop

lea bx,start

sub cx,bx

dec cx

add cx,20

mov bx,cs

mov dx,ds

sub bx,dx

sal bx,1

sal bx,1

sal bx,1

sal bx,1

add cx,bx

push ds

pop bx

int 70h

;===============================================

mov ax, 4C00h

int 21h

eop:

start endp

cseg ends

end start

Далее нужно снова получить файл .exe из ассемблерного кода. И затем выполнить команду:

mymymptdtest.exe

Дамп исполняемого кода в памяти:

OPERATIVE MEMORY DAMP

Beginning address : 4BD2

Number of bytes : 007A

0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0010: B8 D2 4B 8E D8 EB 01 90 8B 0E 00 00 8B 0E 05 00 ёТKЋШл.ђ‹...‹...

0020: 8B 0E FD 00 8B 0E 00 01 8B D9 A1 01 10 8B D8 B9 ‹...‹...‹ЩЎ..‹Ш№

0030: 4C 00 B9 47 00 BB 00 00 2B CB 49 83 C1 14 8C CB L.№G.»..+ЛIѓБ.ЊЛ

0040: 8C DA 2B DA D1 E3 D1 E3 D1 E3 D1 E3 03 CB 1E 5B ЊЪ+ЪСгСгСгСг.Л.[

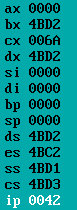
0050: CD 70 CC 00 4C CD 21 00 00 00 00 00 00 00 00 00 НpМ.LН!.........

0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0070: 00 00 00 00 00 00 00 00 00 00 00 -- -- -- -- --

Сегментный адрес распечатываемой области памяти равен 4BD2; он соответствует сегменту данных, имеющему относительный адрес 0000.Это же значение (в остроконечном формате) находится в команде B8 D2 4B (movax,dseg в исходном модуле на ассемблере). Вспомним, что после компоновщика эта команда выглядела так: B8 00 00. Величина 00 00 меньше 4BD2 на 4BD2, которая и равна адресу загрузки программы test.exe, учитывая, что перед сегментом данных нет сегмента стека.

Значение регистров CSи SS:



CS = 4BD3

SS = 4BD1

Следовательно, регистры загружены корректно.

5. Вывод

Я познакомился со структурой обьектного и исполняемого файла, а также проанализировал структуру объектного и загрузочного файла.