

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе № 6
по дисциплине «Операционные системы»
Тема: «Построение модуля динамической структуры»

Студент гр. 8381

Сахаров В.М.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. В отличие от предыдущих лабораторных работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС.

В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4B00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

Постановка задачи:

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет функции:

1. Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.
2. Вызываемый модуль запускается с использованием загрузчика.
3. После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

В качестве вызываемой программы необходимо взять программу ЛР 2, которая распечатывает среду и командную строку. Эту программу следует

немного модифицировать, вставив перед выходом из нее обращение к функции ввода символа с клавиатуры. Введенное значение записывается в регистр AL и затем происходит обращение к функции выхода 4Ch прерывания int 21h.

Шаг 2. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введите произвольный символ из числа A-Z. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 3. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введите комбинацию символов Ctrl+C. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 4. Запустите отлаженную программу, когда текущим каталогом является какой-либо другой каталог, отличный от того, в котором содержатся разработанные программные модули.

Повторите ввод комбинаций клавиш. Занесите полученные данные в отчет.

Шаг 5. Запустите отлаженную программу, когда модули находятся в разных каталогах. Занесите полученные данные в отчет.

Необходимые сведения для составления программы.

Для загрузки и выполнения одной программы из другой используется функция 4B00h прерывания int 21h (загрузчик ОС). Перед обращением к этой функции необходимо выполнить следующие действия:

1. Подготовить место в памяти. При начальном запуске программы ей отводится вся доступная в данный момент память 0S, поэтому необходимо освободить место в памяти. Для этого можно использовать функцию 4Ah прерывания int 21h. Эта функция позволяет уменьшить

отведенный программе блок памяти. Перед вызовом функции надо определить объем памяти, необходимый программе ЛР 6 и задать в регистре ВХ число параграфов, которые будут выделяться программе. Если функция 4Ah не может быть выполнена, то устанавливается флаг переноса СР=1 и в АХ заносится код ошибки:

7 — разрушен управляющий блок памяти;

8 — недостаточно памяти для выполнения функции;

9 — неверный адрес блока памяти;

Поэтому после выполнения каждого прерывания int 21h следует проверять флаг переноса СР=1.

2. Создать блок параметров. Блок параметров - это 14-байтовый блок памяти, в который помещается следующая информация:

dw — сегментный адрес среды

dd — сегмент и смещение командной строки

dd — сегмент и смещение первого FCB

dd — сегмент и смещение второго FCB

Если сегментный адрес среды 0, то вызываемая программа наследует среду вызывающей программы. В противном случае вызывающая программа должна сформировать область памяти в качестве среды, начинающуюся с адреса кратного 16 и поместить этот адрес в блок параметров.

Командная строка записывается в следующем формате:

Первый байт — счетчик, содержащий число символов в командной строке, затем сама командная строка, содержащая не более 128 символов.

На блок параметров перед загрузкой вызываемой программы должны указывать ES:BX.

3. Подготовить строку, содержащую путь и имя вызываемой программы. В конце строки должен стоять код ASCII 0. На подготовленную строку должны указывать DS:DX.

4. Сохранить содержимое регистров SS и SP в переменных. При восстановлении SS и SP нужно учитывать, что DS необходимо также восстановить.

Код завершения формируется вызываемой программой в регистре AL перед выходом в OS с помощью функции 4Ch прерывания int 21h.

В качестве вызываемой программы целесообразно использовать программу, разработанную в Лабораторной работе №2, модифицировав ее следующим образом. Перед выходом из программы перед выполнением функции 4Ch прерывания int 21h следует запросить с клавиатуры символ и поместить введенный символ в регистр AL, в качестве кода завершения. Это можно сделать с помощью функции 01h прерывания int 21h.

Введенный символ остается в регистре AL и служит аргументом для функции 4Ch прерывания int 21h.

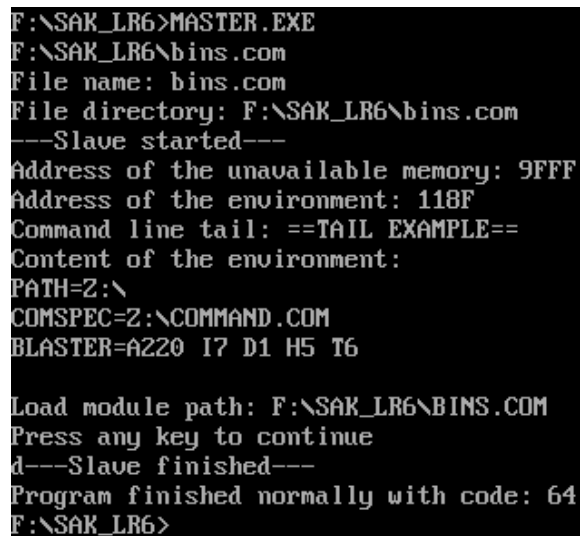
Описание программы.

В результате выполнения лабораторной работы была написана программа, описание функций которой представлено ниже.

- HEX_BYTE_PRINT - выводит на экран один байт в 16 системе счисления, принимая его в регистре AL;
- PRINT_STRING - вывод строки из DX на экран;
- PRINT_LINE - вывод строки из DX на экран с переносом строки;
- FREE_MEMORY - освобождение незанятого программой места в памяти;
- FINISH - вывод сообщения на экран в зависимости от завершения вложенной программы;
- MAKE_PATH - формирование пути к текущему каталогу и пути к исполняемому модулю (регистр AX);

Ход работы

Написание исходного кода производилось в редакторе vscode на базе операционной системы Windows 10, сборка и отладка производились в эмуляторе DOSBox. Поскольку эмуляторы DOS не поддерживают прерывания по Ctrl+Break, отладка обработки этого прерывания проводилась в системе Windows XP.



```
F:\SAK_LR6>MASTER.EXE
F:\SAK_LR6\bins.com
File name: bins.com
File directory: F:\SAK_LR6\bins.com
---Slave started---
Address of the unavailable memory: 9FFF
Address of the environment: 118F
Command line tail: ==TAIL EXAMPLE==
Content of the environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Load module path: F:\SAK_LR6\BINS.COM
Press any key to continue
d---Slave finished---
Program finished normally with code: 64
F:\SAK_LR6>
```

Рисунок 1 — Выполнение программы с прерыванием по введённой букве «d»

Как видно из рисунка, программа завершилась в штатном режиме с кодом 64.

```

C:\ASM>MASTER.EXE
C:\ASM\bins.com
File name: bins.com
File directory: C:\ASM\bins.com
---Slave started---
Address of the unavailable memory: 9FFF
Address of the environment: 14F8
Command line tail: ==TAIL EXAMPLE==
Content of the environment:
COMSPEC=C:\WINDOWS.0\SYSTEM32\COMMAND.COM
ALLUSERSPROFILE=C:\DOCUME~1\ALLUSE~1.0
APPDATA=C:\DOCUME~1\username\APPLIC~1
CLIENTNAME=Console
COMMONPROGRAMFILES=C:\PROGRA~1\COMMON~1
COMPUTERNAME=COMPUTER-5BFD55
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
HOMEPATH=\Documents and Settings\username
LOGONSERVER=\\COMPUTER-5BFD55
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
PATH=C:\WINDOWS.0\system32;C:\WINDOWS.0;C:\WINDOWS.0\System32\Wbem
PATHEXT=.COM;.EXE;.BAT;.CMD;.UBS;.UBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 158 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=9e09
PROGRAMFILES=C:\PROGRA~1
PROMPT=$P$G
SESSIONNAME=Console
SYSTEMDRIVE=C:
SYSTEMROOT=C:\WINDOWS.0
TEMP=C:\WINDOWS.0\TEMP
TMP=C:\WINDOWS.0\TEMP
USERDOMAIN=COMPUTER-5BFD55
USERNAME=username
USERPROFILE=C:\DOCUME~1\username
BLASTER=A220 I5 D1 P330 T3

Load module path: C:\ASM\bins.com
Press any key to continue
^C
---Slave finished---
Program stopped by CTRL+C command
C:\ASM>_

```

Рисунок 2 – Выполнение программы с прерыванием по введённой комбинации «Ctrl+C»

```

F:\SAK_LR6>MASTER.EXE
F:\SAK_LR6\bins.com
File name: bins.com
File directory: F:\SAK_LR6\bins.com
---Slave started---
---Slave finished---
File not found in: F:\SAK_LR6\bins.com
F:\SAK_LR6>

```

Рисунок 3 — Программа и модули находятся в разных каталогах

Как видно из рисунка, модуль найден не был, программа bins.com.exe не была запущена.

Вывод.

В результате выполнения данной лабораторной работы была изучена возможность встраивания пользовательского обработчика прерываний от клавиатуры в стандартный.

Контрольные вопросы.

Как реализовано прерывание Ctrl+C:

Когда нажата комбинация Ctrl+C, DOS вызывает прерывание int 23h. Уровень чувствительности к этому прерыванию может быть проверен и установлен функцией 33h прерывания int 21h.

В какой точке заканчивается вызываемая программа, если код завершения 0:

В месте вызова функции 4Ch прерывания int 21h.

В какой точке заканчивается вызываемая программа по прерыванию Ctrl+Break:

В точке вызова функции 01h прерывания int 21h, где ожидался ввод из клавиатуры.