# Московский государственный технический университет им. Н.Э. Баумана

|  |
| --- |
| Утверждаю: |
| Большаков С.А. |
| " " 2020 г. |

Курсовая работа по курсу «Системное программирование»

# «Резидентная программа (TSR)»

Техническое описание (вид документа)

писчая бумага (вид носителя)

6

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студенты группы ИУ5-41  Алехин С.С. |  |
| " " 2020 г. |

Москва – 2020

# СОДЕРЖАНИЕ

1. [ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ ОБЕСПЕЧЕНИИ 3](#_bookmark0)
2. [МОДУЛЬНАЯ СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. 3](#_bookmark1)
3. [ОПИСАНИЕ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. 3](#_bookmark2)
4. [ДАННЫЕ И ФАЙЛЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 3](#_bookmark3)
5. [ОСНОВНЫЕ АЛГОРИТМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. 4](#_bookmark4)
6. [ПРОЦЕДУРЫ И ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 5](#_bookmark5)
7. [ВЕКТОРА ПРЕРЫВАНИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПЕРЕОПРЕДЕЛЯЕМЫЕ В ПРОГРАММЕ 6](#_bookmark6)

# ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ ОБЕСПЕЧЕНИИ.

*Исходный код, язык*: Assembler *Компилятор*: Turbo Assembler Version 4.1 *Сборщик*: Turbo Link Version 7.1.30.1 *Отладчик*: Turbo Debugger Version 5.0 *Исполняемый код*: файл tsr.com (1 744 байт) *Исходный код*: файл tsr.asm (16 898 байт)

# МОДУЛЬНАЯ СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

Программа делится на резидентную и нерезидентную (инициализирующую части).

Резидентная часть реализует функционал данного программного обеспечения, а нерезидентная нужна для инициализации резидентной части и для обработки параметров командной строки.

См. документ «Модульная структура программы».

# ОПИСАНИЕ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

См. документ «Схема взаимодействие с аппаратурой»

# ДАННЫЕ И ФАЙЛЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Модуль tsr.asm

|  |  |  |
| --- | --- | --- |
| **Имя** | **Размер** | **Хранящиеся данные** |
| ignoredRussianChars | 1 байт | Список игнорируемых символов |
| ignoredRussianCharsLenght | 1 байт | Длина строки ignoredRussianChars |
| isIgnoredRussianChars | 1 байт | Флаг функции игнорирования ввода |
| translateStrFrom | 1 байт | Символы для перевода |
| translateStrTo | 1 байт | Символ, на которые будет идти замена |
| translateSirFromLength | 1 байт | Длина строки translateStrFrom |
| isTranslateStr | 1 байт | Флаг функции перевода |
| signatureFormDelay | 1 байт | Задержка перед выводом подписи |
| signatureCounter | 2 байта | Счетчик подписи |
| isSignatureForm | 1 байт | Флаг функции вывода информации об авторе |
| signatureLine1 | 1 байт | Первая строка подписи |
| signatureLine1Length | 1 байт | Длина первой строки |
| signatureLine2 | 1 байт | Вторая строка подписи |
| signatureLine2Length | 1 байт | Длина второй строки |
| signatureLine3 | 1 байт | Третья строка подписи |
| signatureLine3Length | 1 байт | Длина третьей строки подписи |
| tableTop | 1 байт | Верх таблицы подписи |
| tableTopLength | 1 байт | Длина верха таблицы |
| tableBottom | 1 байт | Низ таблицы подписи |
| tableBottomLength | 1 байт | Длина низа таблицы |
| charToCursiveIndex | 1 байт | Символ для замены |
| cursiveSymbol | 1 байт | Курсивный вариант символа для замены |
| isCursiveChar | 1 байт | Флаг перевода функции в курсив |
| savedSymbol | 1 байт | Переменная для хранения старого символа |
| true | 1 байт | Константа истинности |
| old\_int9hOffset | 2 байта | Адрес старого обработчика int 9h |
| old\_int9hSegment | 2 байта | Сегмент старого обработчика int 9h |
| old\_int1ChOffset | 2 байта | Адрес старого обработчика int 1Ch |
| old\_int1ChSegment | 2 байта | Сегмент старого обработчика int 1Ch |
| old\_int2FhOffset | 2 байта | Адрес старого обработчика int 2Fh |
| old\_int2FhSegment | 2 байта | Сегмент старого обработчика int 2Fh |
| unloadTSR | 1 байт | 1 – выгрузить резидент |
| notLoadTSR | 1 байт | 1 – не загружать резидент |
| helpMsg | 1 байт | справка в БНФ |
| helpMsgLength | 1 байт | Длина справки |
| errorParamMsg | 1 байт | Сообщение о неправильных параметрах ком. строки |
| errorParamMsgLength | 1 байт | Длина сообщения |
| installedMsg | 1 байт | Сообщение о успешной загрузке резидента |
| alreadyInstalledMsg | 1 байт | Сообщение о уже загруженном резиденте |
| noMemMsg | 1 байт | Сообщение о нехватке памяти |
| notInstalledMsg | 1 байт | Сообщение о не успешной загрузке резидента |
| removedMsg | 1 байт | Сообщение о выгрузке резидента |
| removedMsgLength | 1 байт | Длина сообщения |
| noRemoveMsg | 1 байт | Сообщение о не успешной выгрузке резидента |
| noRemoveMsgLength | 1 байт | Длина сообщения |

* 1. **ОСНОВНЫЕ АЛГОРИТМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.**

**Ход инициализации:**

Устанавливается требуемый видеорежим для вывода текстовых сообщений во время работы резидента, вызывается обработчик параметров командной строки, затем, если программа запущена без параметров, то происходит установка резидента и удаление из ОП кода ниже метки INIT\_TSR, если же задан флаг /? выводит справка по работе с программой.

## Обработчик new\_int2Fh:

Вначале проверяется, равен ли регистр AH 0FFh, если это так, то эта наша подфункция, и при AL=0 мы заносим в AH ‘i’, как признак того, что резидент уже загружен в память, а при AL=1 выполняется выгрузка резидента из памяти.

## Обработчик new\_int9h:

Из порта достаётся скан-код нажатой клавиши, по Ctrl+u/U резидент выгружается, по функциональным клавишам (F1, F2, F8, F9) меняют свои значения флаги соответствующих функционалов, затем вызывается стандартный обработчик данного прерывания, если введённый символ входит в множество ограничения ввода (ignoredRussianChars), он блокируется и не выводится, если входит в множество символов под перевод (translateStrFrom) – заменяется на символ с тем же индексом из множества (translateStrTo).

## Обработчик new\_int1Ch:

В самом начале работы обработчика производится вызов старого обработчика прерывания INT 1CH. В случае, если флаг isSignatureForm установлен в true производится сравнение счетчика signatureCounter вызовов прерывания системой с числом signatureFormDelay \*1000/55 + 1, где signatureFormDelay – число в секундах. Если эти числа равны, то далее производится печать информации об исполнителях курсовой работы на экран; иначе signatureCounter увеличивается на 1.

## Функция вывода подписи на экран (printSignature):

## Читается текущее положение курсора на экране и запоминается в стеке. Далее происходит установка положения подписи на экране (верх). Затем эта информация используется для вывода построчно подписи, содержащей верх рамки, три строки собственно информации об исполнителях и низ рамки. Восстанавливается положение курсора из стека.

## Функция проверки командной строки (commandParamsParser):

В регистр SI помещается смещение 80h. Читается количество символов в параметрах командной строки. Если введенный параметр является «/?», то выводится справка по использованию программы и устанавливается флаг того, что загружать резидент не надо. Если параметр не равен «/?», то выводится сообщение о ошибке параметров командной строки.

## Функция получения текущего изображения символа (saveFont):

В стек сохраняются регистры AX и BX. В AX заносится параметр 1130h, в BX – 0600h. Это необходимо, чтобы была вызвана нужная подфункция прерывания 10h. Затем восстанавливаются регистры AX и BX. В результате выполнения функции, регистр ES получает значение C000h, а по адресу ES:BP находятся первый символ таблицы изображений символов, где на каждый символ отводится по 16 байт.

## Функция замены изображения символа (changeFont):

В стек сохраняются регистры AX и BX. В AX заносится параметр 1100h, в BX – 1000h. Это необходимо, чтобы была вызвана нужная подфункция прерывания 10h. Затем восстанавливаются регистры AX и BX. В результате выполнения функции, начиная с номера символа, указанного в регистре DL, изображения символов, количество которых определено в регистре CL, меняется на изображения из таблицы, заданной по адресу ES:BP.

## Функция, меняющая изображение символа с курсива на обычное и наоборот (setCursive):

В стек сохраняются регистры AX, в регистр ES загружается значение регистра CS. Далее, если флаг isCursiveChar установлен в true происходит сохранение текущего изображения изменяемого символа и последующая замена на новое. Номер изменяемого символа содержит переменная cursiveSymbol. С помощью процедуры saveFont определяется адрес текущей таблицы символов. Затем добавляя необходимое значение к регистру BP (16\* cursiveSymbol) находим адрес нужного символа и сохраняем 16 байт таблицы его изображения в переменную savedSymbol. После чего в регистр CX заносится

1 (меняем один символ), в DL устанавливается номер изменяемого символа, в BP перемещается адрес таблицы нового символа. Сама таблица находится в переменной cursiveSymbol. Происходит вызов функции changeFont. Далее выход из процедуры. Если флаг IsCursiveChar не установлен в true, то происходит восстановление старого изображения символа. В регистр CX устанавливается 1, в DL - номер изменяемого символа, в BP перемещается адрес таблицы старого символа (адрес переменной savedSymbol). После чего происходит вызов функции changeFont и завершение процедуры.

См. документ «Блок-схема алгоритма программы».

# ПРОЦЕДУРЫ И ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные данные | Выходные  данные | Описание |
| new\_int9h | - | - | Обработчик  прерывания 09h |
| new\_int1Ch | - | - | Обработчик  прерывания 1Ch |
| new\_int2Fh | AH = 0FFh  AL = 1; для выгрузки TSR AL = 0; для проверки факта  присутствия TSR в памяти | AH = ‘i’, если  резидент присутствует в памяти | Обработчик  прерывания 2Fh |
| printSignature | - | - | Процедура вывода подписи (ФИО, группа,  вариант) |
| commandParamsParser | - | - | Процедура проверки параметров командной  строки |
| saveFont | BH - тип возвращаемой символьной таблицы  0 - таблица из int 1fh 1 - таблица из int 44h  2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14  6 - 8x16 | в ES:BP  находится таблица символов (полная)  в CX находится байт на символ в DL количество экранных строк | Функция сохранения нормального начертания символа |
| changeFont | DL = номер символа для замены  CX = Количество символов заменяемых изображений символов (начиная с символа указанного в DX)  ES:BP = адрес таблицы | - | Функция смены начертания символа |
| setCursive | - | - | Процедура, которая в зависимости от флага IsCursiveChar меняет начертание символа с курсива на обычное и  наоборот |

# ВЕКТОРА ПРЕРЫВАНИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПЕРЕОПРЕДЕЛЯЕМЫЕ В ПРОГРАММЕ.

В программе переопределяются 3 вектора прерываний:

* + 1. 09h – для обработки нажатия клавиш,
    2. 2Fh – для возможности проверки наличия программы в памяти, а также для выгрузки резидентной части программы,
    3. 1Ch – для подсчёта количества времени, прошедшего с нажатия функциональной клавиши, для последующего вывода сообщения-подписи на экран.