

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

Утверждаю:

Большаков С.А.

"\_\_" \_\_\_\_\_ 2023 г.

**Курсовая работа по курсу «Системное программирование»  
«Резидентная программа (TSR)»**

Техническое описание  
(вид документа)

писчая бумага  
(вид носителя)

7  
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-45Б  
Коновалов И.Н.

\_\_\_\_\_  
"\_\_" \_\_\_\_\_ 2023 г.

## СОДЕРЖАНИЕ

1.	ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ ОБЕСПЕЧЕНИИ. ....	3
2.	МОДУЛЬНАЯ СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	3
3.	ОПИСАНИЕ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ....	3
4.	ДАННЫЕ И ФАЙЛЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	3
5.	ОСНОВНЫЕ АЛГОРИТМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ....	4
6.	ПРОЦЕДУРЫ И ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ .....	6
7.	ВЕКТОРА ПРЕРЫВАНИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПЕРЕОПРЕДЕЛЯЕМЫЕ В ПРОГРАММЕ. ....	7

## 1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ ОБЕСПЕЧЕНИИ.

*Исходный код, язык:* Assembler

*Компилятор:* Turbo Assembler Version 3.7

*Сборщик:* Turbo Link Version 7.1.30.1

*Отладчик:* Turbo Debugger Version 5.0

*Исполняемый код:* файл tsr2.com

*Исходный код:* файл tsr2.asm

## 2. 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».

## 3. ОПИСАНИЕ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

См. документ «Схема взаимодействие с аппаратурой»

## 4. ДАННЫЕ И ФАЙЛЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### Модуль tsr.asm

Имя	Размер	Хранящиеся данные
ignoredChars	1 байт	список игнорируемых символов
replaceWith	1 байт	Список символов для замены
ignoredLength	1 байт	длина строки ignoredChars
ignoreEnabled	1 байт	флаг функции игнорирования ввода
translateFrom	1 байт	символы для замены
translateTo	1 байт	символы на которые будет идти замена
translateLength	1 байт	длина строки trasnlateFrom
translateEnabled	1 байт	флаг функции перевода
signaturePrintingEnabled	1 байт	флаг функции вывода информации об авторе
cursiveEnabled	1 байт	флаг перевода символа в курсив
cursiveSymbol	1 байт	символ, составленный из единичек (его курсивный вариант)
charToCursiveIndex	1 байт	символ для замены
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

Имя	Размер	Хранящиеся данные
notLoadTSR	1 байт	1 - не загружать
counter	2 байта	Счётчик
printDelay	1 байт	задержка перед выводом "подписи" в секундах
printPos	2 байта	положение подписи на экране. - верх, 1 - центр, 2 – низ
signatureLine1	1 байт	подпись, строка 1
Line1_length	1 байт	её длина
signatureLine2	1 байт	подпись, строка 2
Line2_length	1 байт	её длина
signatureLine3	1 байт	подпись, строка 3
Line3_length	1 байт	её длина
helpMsg	1 байт	справка в БНФ
helpMsg_length	1 байт	её длина в символах
errorParamMsg	1 байт	сообщение об неверных параметрах ком. строки
errorParamMsg_length	1 байт	его длина
tableTop	1 байт	верх таблицы
tableTop_length	1 байт	его длина
tableBottom	1 байт	низ таблицы
tableBottom_length	1 байт	его длина
installedMsg	1 байт	Сообщение 'Резидент загружен!'
alreadyInstalledMsg	1 байт	Сообщение 'Резидент уже загружен'
noMemMsg	1 байт	Сообщение 'Недостаточно памяти'
notInstalledMsg	1 байт	Сообщение 'Не удалось загрузить резидент'
removedMsg	1 байт	Сообщение 'Резидент выгружен'
removedMsg_length	1 байт	его длина
noRemoveMsg	1 байт	Сообщение 'Не удалось выгрузить резидент'
noRemoveMsg_length	1 байт	его длина
F3_txt	1 байт	строка "F3"
F4_txt	1 байт	строка "F4"
F5_txt	1 байт	строка "F5"
F6_txt	1 байт	строка "F6"
fx_length	1 байт	длина строк fx_txt

## 5. ОСНОВНЫЕ АЛГОРИТМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

### Ход инициализации:

Устанавливается требуемый видеорежим для вывода текстовых сообщений во время работы резидента, вызывается обработчик параметров ком. строки, затем, если программа запущена без параметров, то происходит установка резидента и удаление из ОП кода ниже метки \_initTSR, если же задан флаг /? выводит справка по работе с программой, если задан флаг /u, то, если резидент уже был загружен он выгружается из памяти, восстанавливая при этом старые обработчики прерываний. В коде (закомментировано) предусмотрена также возможность выгрузки резидента по повторному запуску tsr.com.

### 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 резидент выгружается, по функциональным клавишам (F3, ..., F6) меняют свои значения флаги соответствующих функционалов, а также меняется их индикация в верхнем правом углу консоли, затем вызывается стандартный обработчик данного прерывания, если введенный символ входит в множество игнорируемых, он не выводится, если входит в множество символов под замену (translateFrom) – заменяется на символ с тем же индексом из множества translateTo. Вместо игнорирования некоторых символов в некоторых вариантах требуется замена оных на символы другого множества или на звёздочки, всё это предусмотрено в коде, но закоментировано во избежание конфликтов в логике.

#### **Обработчик new\_int1Ch:**

В самом начале работы обработчика производится вызов старого обработчика прерывания int 1Ch. В случае, если флаг signaturePrintingEnabled установлен в true производится сравнение счетчика counter вызовов прерывания системой с числом  $\text{printDelay} * 1000 / 55 + 1$ , где printDelay – число в секундах. Если эти числа равны, то далее производится печать информации об исполнителях курсовой работы на экран; иначе counter увеличивается на 1.

#### **Функция вывода подписи на экран (printSignature):**

Читается текущее положение курсора на экране и запоминается в стеке. Далее происходит выбор положения подписи на экране (верх, центр или низ). В каждом случае устанавливаются значения регистров DH и DL, хранящие информацию о строке и колонке соответственно. 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. Вызывается функция changeFx для прекращения индикации вывода подписи.

#### **Функция вывода индикации:**

Читается текущее положение курсора на экране и запоминается в стеке. Далее происходит перебор всех четырех флагов состояний {signaturePrintingEnabled, cursiveEnabled, translateEnabled, ignoreEnabled}, и, в случае, если флаг установлен в true, то происходит печать наименования функциональной клавиши соответствующего флага состояния в правом верхнем углу экрана на зеленом фоне, иначе на красном. Восстанавливается положение курсора из стека.

#### **Функция проверки командной строки:**

В регистр SI помещается смещение 80h. Читается количество символов в параметрах командной строки. Если их количество равно 0, то выходим. Далее идет цикл до тех пор, пока не будет прочитан символ возврата каретки. Если одним из параметров является «/?» то выводится справка по использованию программы и устанавливается флаг того, что загружать резидент не надо (notLoadTSR).

#### **Функция получения текущего изображения символа:**

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:**

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:**

В стек сохраняются регистры AX, в регистр ES загружается значение регистра CS. Далее, если флаг `cursiveEnabled` установлен в `true` происходит сохранение текущего изображения изменяемого символа и последующая замена на новое. Номер изменяемого символа содержит переменная `charToCursiveIndex`. С помощью процедуры `saveFont` определяется адрес текущей таблицы символов. Затем добавляя необходимое значение к регистру BP ( $16 * \text{charToCursiveIndex}$ ) 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 `cursiveEnabled` 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.

Примечание: значения переменных `charToCursiveIndex` и `cursiveSymbol` зависят от варианта.

См. документ «Блок-схема алгоритма программы».

## 6. ПРОЦЕДУРЫ И ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Название	Входные данные	Выходные данные	Описание
<code>new_int9h</code>	-	-	Обработчик прерывания 09h
<code>new_int1Ch</code>	-	-	Обработчик прерывания 1Ch
<code>new_int2Fh</code>	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
<code>printSignature</code>	-	-	Процедура вывода подписи (ФИО, группа, вариант)
<code>setCursive</code>	-	-	Процедура, которая в зависимости от флага <code>cursiveEnabled</code> меняет начертание символа с курсива на обычное и наоборот
<code>changeFont</code>	DL = номер символа для замены CX = Количество символов заменяемых изображений символов (начиная с	-	Функция смены начертания символа

Название	Входные данные	Выходные данные	Описание
	символа указанного в DX) ES:BP = адрес таблицы		
saveFont	ВН - 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
commandParamsParser	-	-	Процедура проверки параметров командной строки
changeFx	-	-	Процедура вывода состояния клавиш Fx

## 7. 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.