Федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА

Программирование на ассемблере

По дисциплине «Архитектура ЭВМ. Часть 1»

Выполнила студентка гр. в5130904/30030 Назарова К.А.

Руководитель
проф. д.т.н. С.А. Молодяков

Санкт-Петербург
2024

Оглавление

Введение 3

Программа 1 4

Блок-схема 4

Список использованных прерываний BIOS и DOS 7

Текст программы 7

Результат выполнения программы 13

Введение

Язык программирования ассемблер является одним из наиболее фундаментальных инструментов в мире компьютерной науки и разработки программного обеспечения. Он представляет собой низкоуровневый язык, тесно связанный с аппаратурой компьютера, что позволяет разработчикам максимально контролировать ресурсы и поведение вычислительной системы. В данной курсовой работе делается упор на изучении основных аспектов языка ассемблера, его возможностях и преимуществах.

Одним из главных преимуществ ассемблера является его высокая эффективность. Благодаря непосредственному управлению аппаратурой компьютера и прямому доступу к ресурсам процессора, программы, написанные на ассемблере, могут быть оптимизированы до максимальной производительности. Это особенно важно в областях, где требуется максимальное быстродействие, таких как системное программирование, разработка драйверов устройств и встраиваемых систем.

Другим важным аспектом ассемблера является его непосредственное воздействие на аппаратное обеспечение компьютера. Разработчики имеют возможность полностью контролировать работу процессора, управлять памятью и вводом-выводом данных, что делает ассемблер незаменимым инструментом при работе с периферийными устройствами и низкоуровневым программированием.

Кроме того, изучение ассемблера позволяет разработчикам глубже понять принципы работы компьютерных систем. Это помогает не только в создании эффективного программного обеспечения, но и в обнаружении и исправлении неполадок на уровне аппаратуры.

В настоящей работе мы рассмотрим основные концепции и инструменты языка ассемблера, его возможности и применение в современной разработке программного обеспечения. Благодаря этому, читатель сможет оценить важность и актуальность изучения данного языка для профессионального роста в области компьютерных наук и разработки программного обеспечения.

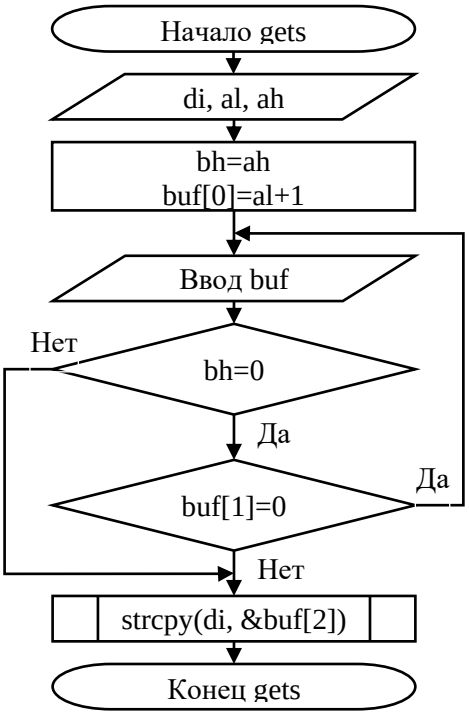
Программа 1

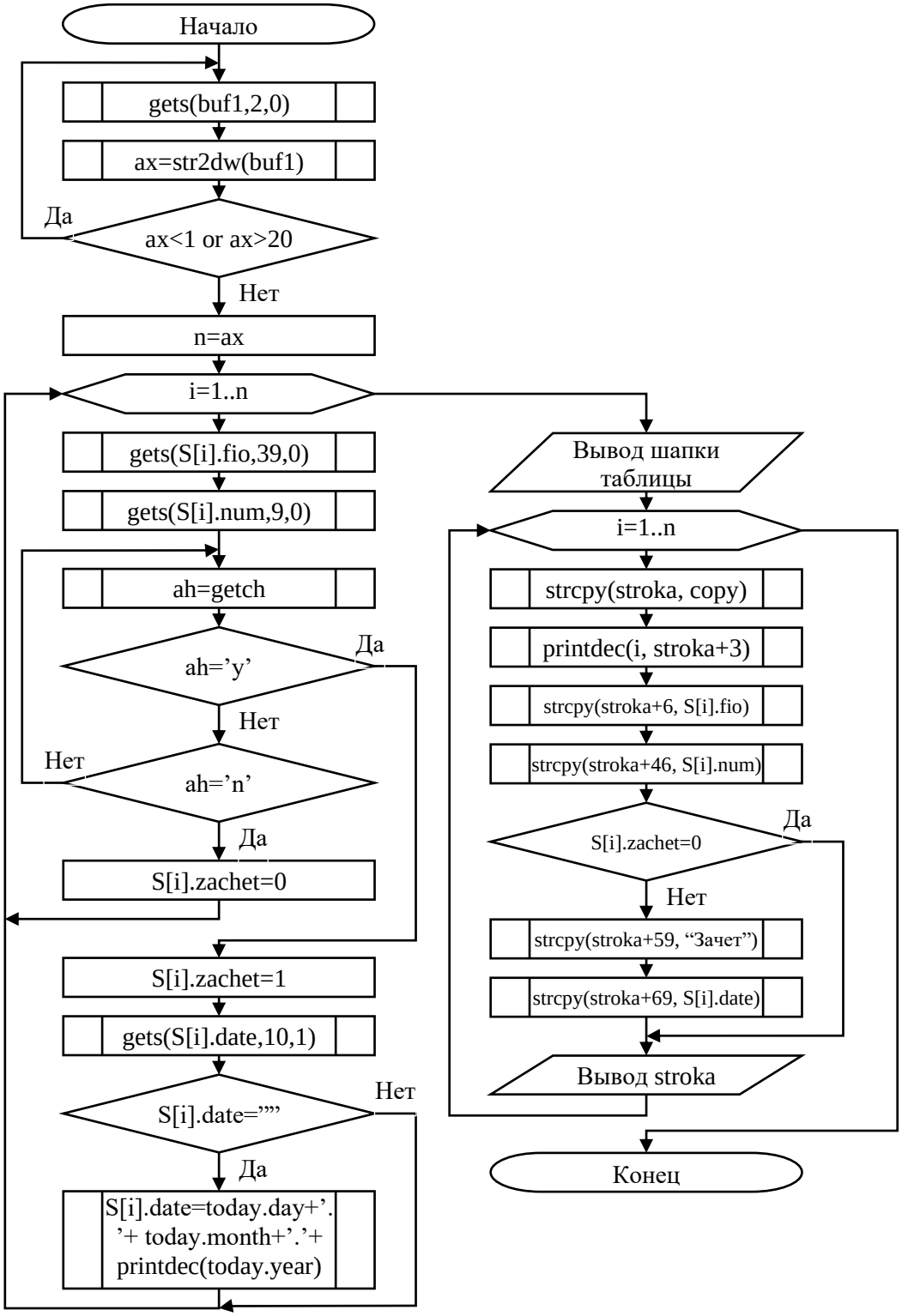
11.Составить "электронную зачетную ведомость" для вашей группы по ТТО:

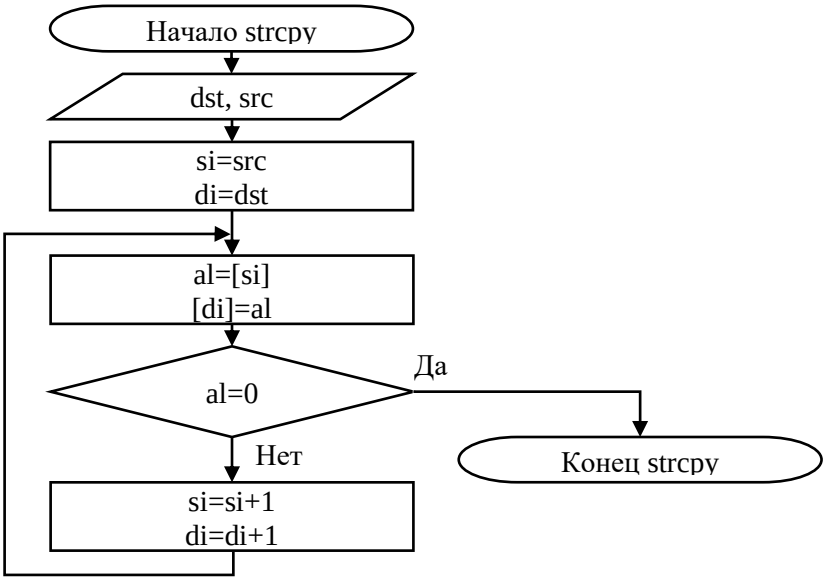
N ¦Фамилия И.О.¦Номер № зачет. книжки ¦Отметка о зачете¦Дата п/п¦ ¦ ¦ ¦

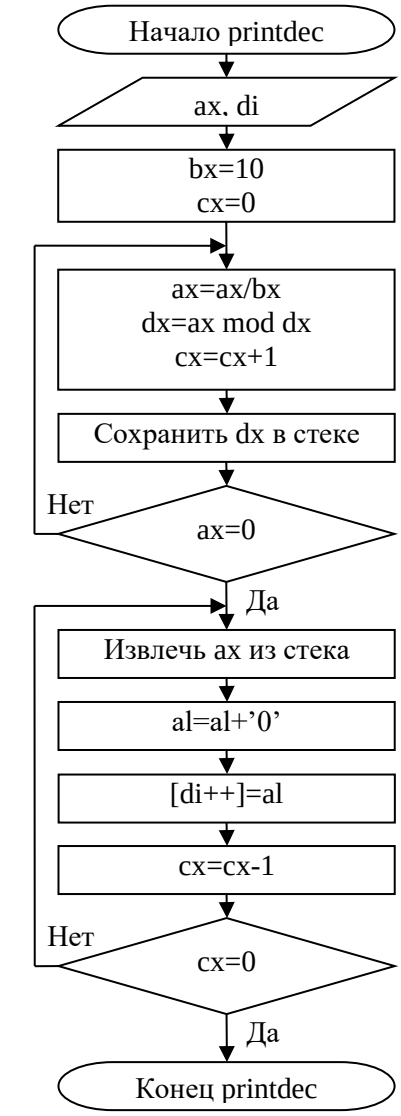
Предусмотреть возможность включения слова "зачет" и даты получения зачета.

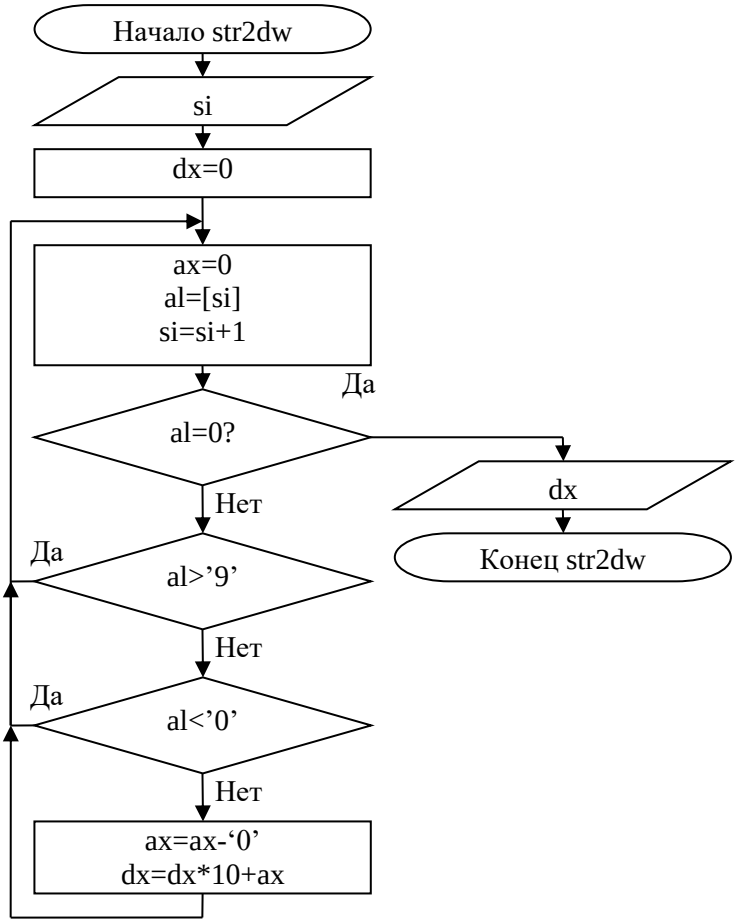
Блок-схема




Начало
gets(buf1,2,0)
ax=str2dw(buf1)
Да
ax<1 or ax>20
Нет
n=ax
i=1..n
gets(S[i].fio,39,0)
gets(S[i].num,9,0)
ah=getch
ah=’y’
Да
Нет
ah=’n’
Нет
Да
S[i].zachet=0
S[i].zachet=1
gets(S[i].date,10,1)
S[i].date=””
Нет
Да
S[i].date=today.day+’.
’+ today.month+’.’+
printdec(today.year)
Вывод шапки
таблицы
i=1..n
strcpy(stroka, copy)
printdec(i, stroka+3)
strcpy(stroka+6, S[i].fio)
strcpy(stroka+46, S[i].num)
S[i].zachet=0
Да
Нет
strcpy(stroka+59, “Зачет”)
strcpy(stroka+69, S[i].date)
Вывод stroka
Конец


Начало strcpy
dst, src
si=src
di=dst
al=[si]
[di]=al
al=0
Да
Нет
Конец strcpy
si=si+1
di=di+1


Начало printdec
ax, di
bx=10
cx=0
ax=ax/bx
dx=ax mod dx
cx=cx+1
Сохранить dx в стеке
Нет
ax=0
Да
Извлечь ax из стека
al=al+'0'
[di++]=al
cx=cx-1
Нет
cx=0
Да
Конец printdec


Начало str2dw
si
dx=0
ax=0
al=[si]
si=si+1
Да
al=0?
Нет
dx
Конец str2dw
Да
al>'9'
Нет
Да
al<'0'
Нет
ax=ax-'0'
dx=dx*10+ax

Список использованных прерываний BIOS и DOS

INT 16h – BIOS keyboard services

AH (номер функции)	Описание
00h	Чтение кода нажатой клавиши из буфера клавиатуры

INT 21h – Main DOS API

AH (номер функции)	Описание
02h	Вывод символа на экран
09h	Вывод строки на экран
0Ah	Ввод строки с клавиатуры
4Ch	Завершение программы

Текст программы

```
.model small,c
ZAP STRUC ;структура записи ведомости
fio db 40 DUP(?)    ;фио
num db 10 dup(?)         ;номер зачетной книжки
zachet db ?              ;флаг получения зачета
date DB 11 DUP(?)   ;дата зачета
ZAP ENDS
puts macro     string         ;вывод строки
     lea  dx,string
     mov  ah,09h
     int  21h
endm
putch     macro     char      ;вывод символа
     mov  dl,char
     mov  ah,2
     int  21h
endm

getch     macro          ;Ожидание нажатия любой клавиши
     mov ah,0
     int 16h
endm
.data
n    dw 0      ;количество студентов в ведомости
```

```
msg1 db 13,10,'Введите количество студентов в ведомости(1..20):
$'
msg2 db 13,10,'Введите ФИО студента: $'
msg3 db 13,10,'Введите номер зачетной книжки: $'
msg4 db 13,10,'Зачет получен (y/n): $'
msg5 db 13,10,'Дата зачета(пустая строка - текущая дата): $'
zach db 'Зачет',0
titl db 13,10,'                          Электронная зачетная
ведомость$'
head db 13,10,'|N | Фамилия И.О.                              |N
зачетки|Отм.о зачете|   Дата    |$'

buf  db 256 dup(0)  ;принимает строку, введенную с клавиатуры
buf1 db 4 dup(0)
copy db 13,10,'|  |                                          |
|           |           |$',0
stroka    db 83 dup(0)
delim     db 13,10,'-----------------------------------------
----------------------------------$'
S ZAP 20 dup (<0>)  ;массив записей со студентами
.stack 256
.code
start:
     mov ax,@data   ;Настраиваем сегментные регистры
     mov ds,ax
     mov es,ax
m1:  puts msg1      ;вывод сообщения
     mov ax,2       ;длина строки
     lea di,buf1         ;куда вводить
     call gets      ;ввод строки
     lea si,buf1         ;Введенный текст
     call str2dw         ;преобразовать в число
     cmp ax,1       ;если меньше 1
     jb m1               ;повторить ввод
     cmp ax,20      ;если больше 20
     ja m1               ;повторить ввод
     mov n,ax       ;запомнить количество студентов в ведомости
     lea bx,S       ;массив записей со студентами
     mov cx,n       ;количество студентов в ведомости
inlp:     puts msg2      ;вывод сообщения
     mov ax,39      ;длина строки
     lea di,[bx].ZAP.fio     ;куда вводить
     call gets               ;ввод ФИО
     puts msg3      ;вывод сообщения
     mov ax,9       ;длина строки
     lea di,[bx].ZAP.num     ;куда вводить
     call gets         ;ввод номера зачетки
     puts msg4      ;вывод сообщения
m2:  getch               ;Ожидание нажатия любой клавиши
     cmp ah,21      ;если y
     jz yes             ;то переход
     cmp ah,49      ;если не n
```

```
     jnz m2              ;то повторить ввод
     putch al       ;вывести введенный символ
     mov [bx].ZAP.zachet,0    ;пометить что зачет не получен
     jmp ne_pust_data    ;продолжить
yes: putch al       ;вывести введенный символ
     mov [bx].ZAP.zachet,1    ;пометить что зачет не получен
     puts msg5      ;вывод сообщения
     mov ah,1       ;признак что можно вводить пустую строку
     mov al,10      ;длина строки
     lea di,[bx].ZAP.date    ;куда вводить
     call gets              ;ввод даты
     cmp byte ptr [di],0 ;если не пустая дата
     jnz ne_pust_data    ;то продолжить
     push cx             ;сохранить регистр
     mov ah,2ah
     int 21h             ;получить текущую дату
     mov al,dl      ;день
     aam            ;2-10 коррекция
     add ax,3030h        ;преобразовать число в ASCII
     xchg ah,al          ;поменять символы местами перед записью
     stosw               ;записать число
     mov al,'.'
     stosb
     mov al,dh      ;месяц
     aam            ;2-10 коррекция
     add ax,3030h        ;преобразовать число в ASCII
     xchg ah,al          ;поменять символы местами перед записью
     stosw               ;записать месяц
     mov al,'.'
     stosb
     mov ax,cx      ;год
     call printdec       ;записать год
     mov al,0
     stosb               ;добавить конец строки
     pop cx              ;восстановить регистр
ne_pust_data:
     add bx, size ZAP    ;перейти к следующей записи ведомости
     dec cx              ;уменьшить счетчик
     jz q1
     jmp inlp       ;пока счетчик не 0 продолжить ввод
q1:  puts titl      ;вывод шапки таблицы
     puts delim
     puts head
     puts delim
     lea bx,S       ;массив записей со студентами
     mov cx,1       ;номер по порядку
lpout:    call strcpy,offset stroka,offset copy  ;перезаписать
формируемую строку из копии
     lea di,stroka+3          ;место в строке куда записывать
     mov ax,cx
     call printdec       ;добавить в строку номер
     lea ax,[bx].ZAP.fio ;фио в массиве
```

```
     call strcpy,offset stroka+6,ax     ;добавить в строку фио
     lea ax,[bx].ZAP.num ;номер зачетки
     call strcpy,offset stroka+46,ax    ;добавить в строку номер
зачетки
     cmp [bx].ZAP.zachet,0    ;если зачет не получен
     jz no_zach          ;то пропустить
     call strcpy,offset stroka+59,offset zach    ;копирование в
строку "зачет"
     lea ax,[bx].ZAP.date          ;дата зачета
     call strcpy,offset stroka+69,ax    ;скопировать в строку
дату
no_zach:
     puts stroka         ;вывести сформированную строку
     add bx,size ZAP          ;перейти к следующей записи
ведомости
     inc cx              ;увеличить номер
     cmp cx,n       ;если вывели всю ведомость
     ja fin              ;то закончить
     jmp lpout      ;продолжить вывод
fin:
     puts delim
     getch          ;Ожидание нажатия любой клавиши
     mov ax,4c00h   ;закончить программу
     int 21h
;преобразование числа из ax в десятичную строку по адресу es:di
;ax - число
;es:di - адрес буфера приемника
printdec proc
     push cx   ;сохраняем регистры
     push dx
     push bx
     mov bx,10 ;основание системы
     xor cx,cx ;в cx будет количество цифр в десятичном числе
@@m1a:    xor dx,dx
     div bx         ;делим число на степени 10
     push dx        ;и сохраняем остаток от деления(коэффициенты
при степенях) в стек
     inc cx         ;увеличиваем количество десятичных цифр числа
     test ax,ax     ;после деления остался 0?
     jnz @@m1a ;если нет, продолжаем
@@m2a:    pop ax         ;взять из стека цифру числа
     add al,'0'     ;преобразовываем цифру в ASCII символ
     stosb          ;сохраняем в буфер
     loop @@m2a     ;все цифры
     pop bx         ;восстанавливаем регистры
     pop dx
     pop cx
     ret       ;выход из подпрограммы
printdec endp
str2dw    proc
;Преобразование строки в число
```

```
;на входе ds:si ссылается на ASCII строку, которую нужно
преобразовать
;на выходе в ax - число.
     push dx        ;сохраняем регистры
     push si
     xor dx,dx ;сумма
@lp1:     xor ax,ax
     lodsb          ;берем символ
     cmp al,0  ;если это конец строки,
     jz @ex         ;то заканчиваем
     cmp al,'9'     ;если это не цифра,
     jnbe @lp1 ;то пропускаем
     cmp al,'0'      ;если это не цифра,
     jb @lp1        ;то пропускаем
     sub ax,'0'     ;получаем цифровое значение
     shl dx,1  ;умножаем сумму на 10
     add ax, dx
     shl dx, 2
     add dx, ax     ;прибавляем текущее значение
     jmp @lp1  ;продолжаем обработку
@ex: mov ax,dx ;помещаем результат в ax
     pop si    ;восстанавливаем регистры
     pop dx
     ret       ;выход их подпрограммы
str2dw    endp

;копирование одной строки в другую
;dst - результирующая строка
;src - исходная строка
strcpy    proc dst:word, src:word

     push si             ;Сохраняем регистры
     push di
     push ax
     mov si,src          ;исходная строка
     mov di,dst          ;результирующая строка
cpylp:    mov al,[si]         ;взять символ из источника
     mov [di],al         ;скопировать в приемник
     cmp al,0       ;если скопированный символ с кодом  0
     jz cpyex       ;если  0 - то строка скопирована полностью
     inc si              ;следующие символы
     inc di
     jmp cpylp      ;то продолжить копирование
cpyex:    pop ax              ;восстановить регистры
     pop di
     pop si
     ret

strcpy endp
;ввод строки
;ds:di - адрес куда вводить строку
;al -  максимальная длина строки
```

```
;ah=0 = нельзя вводить пустую строку,ah=1 - можно вводить пустую
строку
gets proc
     push bx             ;Сохраняем регистры
     push dx
     push ax
     inc al              ;увеличить максимальную длина строки для
символа конца строки
     mov bh,ah      ;Запомнить признак ввода пустой строки
     mov buf[0],al       ;максимальная длина строки
m111:     mov ah,0ah          ;функция ввода строки с клавиатуры
     mov dx,offset buf      ;буфер куда вводить
     int 21h             ;пользователь вводит в текст в buf
     mov bl,buf[1]       ;длина введенной строки
     cmp bh,0       ;если пустую строку можно вводить
     jnz m112       ;пропустить проверки
     cmp bl,0       ;если не пустая строка
     jnz m112       ;то продолжить
     putch 10       ;перейти на новую строку
     jmp m111       ;повторить ввод
m112:     mov bh,0       ;bx=длина введенной строки
     mov buf[bx+2],byte ptr 0 ;добавить 0 в конец введенной
строки
     call strcpy,di,offset buf[2]  ;сохранить наименование
     pop ax              ;восстановить регистры
     pop dx
     pop bx
     ret
gets endp
end start
```

Результат выполнения программы