МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

# Кафедра проектирования информационно-компьютерных систем

Отчет по предмету

Программирование на языке ассемблера

Лабораторная работа №7

**«ЗАГРУЗКА И ВЫПОЛНЕНИЕ ПРОГРАММ. РАБОТА С ПАМЯТЬЮ»**

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы 150501  Смоленский Н.О. | **Проверил:**  Туровец Н.О. |

Минск 2022

**Цель работы**: ознакомиться с загрузкой и выполнением программ, рассмотреть работу с памятью.

**Теоретические сведения:**

1. *Управление памятью.*

При запуске программы в DOS, ей выделяется все доступная память, поэтому доступно использование, например, памяти за концом программы практически до конца сегмента кода. Однако для загрузки других программ из текущей потребуется свободная память и наиболее простой вариант ее получения – сократить выделенный текущей программе блок памяти до минимума с помощью функции DOS 4Ah.

-- Функция DOS 4Ah (INT 21h) – изменить размер блока памяти.

Также доступно выделение и удаление дополнительных блоков памяти:

-- Функция DOS 48h (INT 21h) – выделить блок памяти.

-- Функция DOS 49h (INT 21h) – освободить блок памяти.

2. *Загрузка и выполнение программ.*

Для загрузки и выполнения программ требуется использовать функцию DOS 4Bh (INT 21h) – загрузить и выполнить программу:

Ввод: АН = 4Bh,

AL = подфункции:

AL = 00h — загрузить и выполнить;

AL = 01h — загрузить и не выполнять;

AL = 03h – загрузить как оверлей;

AL = 05h – подготовиться к выполнению (DOS 5.0+).

Вывод: Если CF = 1, то произошла ошибка (в АХ = код ошибки):

-- 02h – файл не найден,

-- 05h – доступ к файлу запрещен,

-- 08h – не хватает памяти,

-- 0Ah – неправильное окружение,

-- 0Bh – неправильный формат.

Если CF = 0, то операция успешно выполнена.

3. *Оверлейные модули.*

Оверлей – это часть исполняемой программы (обычно процедура, хотя это может быть полностью самостоятельная программа со своими сегментами данных и стека), которая по мере необходимости загружается в определенную область памяти. Различные оверлейные модули могут загружаться в одно и то же место, перекрывая предыдущий код, что позволяет экономить память, но снижает быстродействие программы при частых загрузках.

**Код программы:**

Программа *rand.asm*, генерирующая псевдослучайное число в диапазоне (0…255):

*.model tiny*

*.data*

*msg db "Wise pseudorandom number generator says: $"*

*newl db 0Dh,0Ah,'$'*

*num db 3 dup (?),'$'*

*.code*

*start:*

*mov ax,@data*

*mov ds,ax*

*rand\_num:*

*mov ah,2Ch*

*int 21h*

*mov al,dl*

*mul dh*

*to\_string:*

*mov dl,10*

*lea di,num*

*add di,2*

*mov cx,3*

*to\_string\_loop:*

*xor ah,ah*

*div dl*

*xchg al,ah*

*add al,'0'*

*mov [di],al*

*dec di*

*xchg al,ah*

*loop to\_string\_loop*

*mov ah,9*

*lea dx,msg*

*int 21h*

*lea dx,num*

*int 21h*

*lea dx,newl*

*int 21h*

*mov ax, 4c00h*

*int 21h*

*end start*

Основная программа:

*.model small*

*.stack 100h*

*.data*

*num db 0*

*msg1 db "Unable to load and execute program",0Dh,0Ah,'$'*

*msg2 db "Unable to allocate memory",0Dh,0Ah,'$'*

*msg3 db "Invalid input",0Dh,0Ah,'$'*

*success db "Successfully runned "*

*times db 3 dup(?)*

*db " times!",0Dh,0Ah,'$'*

*command\_line db 126 dup(0)*

*file\_name db 125 dup(0)*

*EPB dw 0000*

*dw offset command\_line*

*dw 005Ch,0,006Ch,0*

*.code*

*printStr MACRO str*

*push ax dx*

*mov dx,offset str*

*mov ah,9*

*int 21h*

*pop dx ax*

*ENDM*

*isDelimiter PROC*

*xor ah,ah*

*cmp al,20h*

*jne not\_space*

*mov ah,1*

*not\_space:*

*cmp al,9h*

*jne not\_tab*

*mov ah,1*

*not\_tab:*

*ret*

*isDelimiter ENDP*

*findWord PROC*

*push bp*

*mov bp,sp*

*push ax cx si*

*mov si,[bp+4]*

*handle\_word:*

*skip\_delimiter:*

*mov al,[si]*

*call isDelimiter*

*cmp ah,1*

*je found\_delimiter*

*jmp stop\_reading\_string*

*found\_delimiter:*

*cmp cx,0*

*jne string\_end\_not\_reached*

*jmp stop\_reading\_string*

*string\_end\_not\_reached:*

*inc si*

*dec cx*

*jmp handle\_word*

*stop\_reading\_string:*

*mov bx,si*

*sub bx,[bp+4]*

*skip\_character:*

*mov al,[si]*

*call isDelimiter*

*cmp ah,1*

*jne not\_delimiter*

*jmp word\_read*

*not\_delimiter:*

*cmp cx,0*

*jne can\_read\_more*

*jmp word\_read*

*can\_read\_more:*

*cmp al,0Dh*

*jne not\_the\_end*

*jmp word\_read*

*not\_the\_end:*

*inc si*

*dec cx*

*jmp skip\_character*

*word\_read:*

*mov di,si*

*sub di,[bp+4]*

*cmp bx,di*

*je word\_not\_found*

*dec dx*

*word\_not\_found:*

*cmp cx,0*

*jne not\_end\_of\_string*

*jmp end\_of\_loop*

*not\_end\_of\_string:*

*cmp dx,0*

*jne not\_this\_word*

*jmp end\_of\_loop*

*not\_this\_word:*

*cmp byte ptr [si],0Dh*

*jne not\_stop*

*jmp end\_of\_loop*

*not\_stop:*

*jmp handle\_word*

*end\_of\_loop:*

*pop si cx ax bp*

*ret*

*findWord ENDP*

*readCommandLine PROC*

*push cx si di*

*xor cx,cx*

*mov cl,ds:[80h]*

*mov bx,cx*

*mov si,81h*

*lea di,command\_line*

*rep movsb*

*pop di si cx*

*ret*

*readCommandLine ENDP*

*getFileNameFromCmd PROC*

*push cx bx dx si di*

*xor ax,ax*

*mov dx,1*

*mov cx,bx*

*push offset command\_line*

*call findWord*

*add sp,2*

*cmp dx,0*

*jne file\_name\_not\_found*

*mov cx,di*

*sub cx,bx*

*mov ax,cx*

*mov di,offset file\_name*

*mov si,bx*

*add si,offset command\_line*

*rep movsb*

*file\_name\_not\_found:*

*pop di si dx bx cx*

*ret*

*getFileNameFromCmd ENDP*

*isDigit PROC*

*xor ch,ch*

*cmp cl,30h*

*jge greater\_than\_zero*

*jmp not\_a\_number*

*greater\_than\_zero:*

*cmp cl,39h*

*jle less\_than\_nine*

*jmp not\_a\_number*

*less\_than\_nine:*

*mov ch,1*

*not\_a\_number:*

*ret*

*isDigit ENDP*

*getNumberFromCmd PROC*

*push bx cx dx si*

*mov dx,2*

*mov cx,bx*

*push offset command\_line*

*call findWord*

*add sp,2*

*cmp dx,0*

*jne invalid\_input*

*mov si,10*

*xor ax,ax*

*get\_another\_digit:*

*mov cl,command\_line[bx]*

*cmp bx,di*

*jne not\_end\_of\_number*

*jmp end\_of\_number*

*not\_end\_of\_number:*

*call isDigit*

*cmp ch,1*

*je digit\_read*

*jmp invalid\_input*

*digit\_read:*

*mul si*

*jno not\_overflow*

*jmp invalid\_input*

*not\_overflow:*

*xor ch,ch*

*sub cx,30h*

*add ax,cx*

*jno not\_of*

*jmp invalid\_input*

*not\_of:*

*inc bx*

*jmp get\_another\_digit*

*end\_of\_number:*

*jmp proc\_end*

*invalid\_input:*

*xor ax,ax*

*proc\_end:*

*pop si dx cx bx*

*ret*

*getNumberFromCmd ENDP*

*toString PROC*

*push ax cx dx di*

*to\_string:*

*mov dl,10*

*lea di,times*

*add di,2*

*mov cx,3*

*to\_string\_loop:*

*xor ah,ah*

*div dl*

*xchg al,ah*

*add al,'0'*

*mov [di],al*

*dec di*

*xchg al,ah*

*loop to\_string\_loop*

*pop di dx cx ax*

*ret*

*toString ENDP*

*allocateMemory PROC*

*push ax bx dx*

*mov ax,es*

*mov bx,zseg*

*sub bx,ax*

*mov ah,4Ah*

*int 21h*

*jnc memory\_allocated*

*printStr msg2*

*memory\_allocated:*

*pop dx bx ax*

*ret*

*allocateMemory ENDP*

*loadProgram PROC*

*push ax bx dx*

*mov ax,ds*

*mov word ptr EPB+4,ax*

*mov word ptr EPB+8,ax*

*mov word ptr EPB+0Ch,ax*

*mov dx,offset file\_name*

*mov bx,offset EPB*

*mov ah,4Bh*

*mov al,0*

*int 21h*

*jnc loaded\_successfully*

*printStr msg1*

*loaded\_successfully:*

*pop dx bx ax*

*ret*

*loadProgram ENDP*

*delay PROC*

*push ax cx dx*

*mov cx,3*

*mov dx,22BCh*

*mov ah,86h*

*int 15h*

*pop dx cx ax*

*ret*

*delay ENDP*

*start:*

*call allocateMemory*

*mov ax,@data*

*mov es,ax*

*call readCommandLine*

*mov ds,ax*

*call getFileNameFromCmd*

*cmp ax,0*

*je invalid\_parameter*

*call getNumberFromCmd*

*mov bx,ax*

*cmp ax,0*

*je invalid\_parameter*

*cmp ax,255*

*jg invalid\_parameter*

*mov num,al*

*call toString*

*xor ch,ch*

*mov cl,num*

*mov al,0*

*iteration:*

*call delay*

*call loadProgram*

*jc \_end*

*loop iteration*

*printStr success*

*\_end:*

*jmp skip\_output\_msg*

*invalid\_parameter:*

*printStr msg3*

*skip\_output\_msg:*

*mov ax,4c00h*

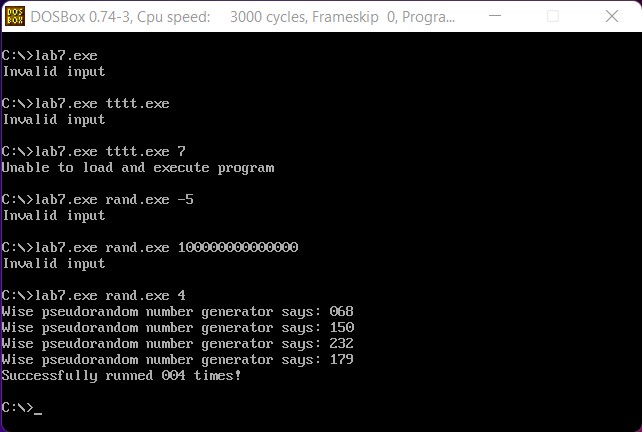
*int 21h*

*zseg SEGMENT*

*zseg ENDS*

*end start*

**Результат работы:**

****

**Вывод**

В ходе выполнения лабораторной работы было проведено ознакомление с загрузкой и выполнением программ, рассмотрена работа с памятью.