Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 7

по дисциплине «Программирование на языке Ассемблера»

Вариант 2

Выполнил студент гр. 150502: Альхимович Н.Г.

Проверил:        Туровец Н.О.

Минск 2022

Цель работы:

Ознакомиться с загрузкой и выполнением программ, рассмотреть работу с памятью.

Вариант задания:

Написать программу, запускающую другую программу N раз (N – число в диапазоне [1, 255]). Имя запускаемой программы передается в командной строке.

Теоретические сведения:

1. Управление памятью.

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

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

Ввод:

АН = 4Ah,  
ВХ =новыйразмерв16-байтныхпараграфах,  
ES = сегментный адрес модифицируемого блока.

Вывод:

если CF = 1, то есть ошибки (в AX – код ошибки):  
-- 07 – блоки управления памятью разрушены,

--08–не хватает памяти(при увеличении, в ВХ = максимальный размер, доступный для этого блока),

-- 09 – ES содержит неверный адрес.

Если CF = 0, то операция выполнена успешно.

Также доступно выделение и удаление дополнительных блоков памяти:

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

Ввод: АН = 48h  
ВХ = размер блока в 16-байтных параграфах.  
Эта функция с ВХ = FFFFh используется для определения размера самого большого доступного блока памяти.

Вывод:

если CF = 1, то есть ошибки (в AX – код ошибки):  
-- 07 – блоки управления памятью разрушены,  
-- 08 – не хватает памяти (ВХ = размер максимального доступного блока).  
Если CF = 0, то операция выполнена успешно, АХ = сегментный адрес выделенного блока.

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

Ввод:

АН = 49h  
ES = сегментный адрес освобождаемого блока

Вывод:

если CF = 1, то есть ошибки (в AX – код ошибки):

-- 07 – блоки управления памятью разрушены,  
-- 09 – ES содержит неверный адрес.

Если CF = 0, то операция выполнена успешно.

2. Загрузка и выполнение программ.

Для загрузки и выполнения программ требуется использовать функцию DOS4Bh(INT 21h) – загрузить и выполнить программу:

Ввод:

АН = 4Bh,  
AL = подфункции:  
AL = 00h — загрузить и выполнить;

AL = 01h—загрузить и не выполнять:

DS:DX – адрес ASCIZ-строки с полным именем программы,

ES:BX – адрес блока параметров ЕРВ:

+00h (слово) – сегментный адрес окружения, которое будет скопировано для нового процесса (или 0, если используется текущее окружение);  
+02h (4 байта) – адрес командной строки для нового процес- са;

+06h (4 байта) – адрес первого FCB для нового процесса;

+0Ah (4 байта) – адрес второго FCB для нового процесса;

+0Eh (4 байта) – здесь будет записан SS:SP нового процесса после его завершения (только для AL = 01);

+12h (4 байта) – здесь будет записан CS:IP (точка входа) нового процесса после его завершения (только для AL = 01).

AL = 03h – загрузить как оверлей:

DS:DX — адрес ASCIZ-строки с полным именем программы,

ES:BX — адрес блока параметров:

+00h (слово) – сегментный адрес для загрузки оверлея,  
+02h (слово) – число, которое будет использовано в командах, использующих непосредственные сегментные адреса (обычно то же самое число, что и в предыдущем поле или 0 для com-файлов).  
AL = 05h – подготовиться к выполнению (DOS 5.0+):

DS:DX – адрес следующей структуры:

+00h(слово) – 00h,  
+02h (слово) :

бит 0 – exe-программа,

бит 1 – программа-оверлей,  
+04h (4 байта) – адрес ASCIZ-строки с именем новой про- граммы;  
+08h (слово) – сегментный адрес PSP новой программы; +0Ah (4 байта) – точка входа новой программы;  
+0Eh (4 байта) – размер программы, включая PSP.

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: ВХ и DX изменены.

Особенности: Для подфункций 00 и 01 требуется, чтобы было достаточно сво- бодной памяти для загрузки программы, поэтому com-программы должны воспользоваться функцией DOS 4Ah для уменьшения отведенного им блока памяти до минимально необходимого.

При вызове подфункции 03, DOS загружает оверлей в память, выделенную текущим процессом, так что exe-программы долж- ны убедиться, что ее достаточно.  
Эта функция игнорирует расширение файла и различает exe- и com-файлы по первым двум байтам заголовка («MZ» для exe- файлов).

Фрагмент программы, демонстрирующий как выполнить загрузку и за- пуск другой программы, приведен ниже.

; вызов программы test.exe

.model tiny

.code

.186

org 100h

...

; СОМ-программа

start:

; перемещение стека на 200h после конца программы

mov sp,program\_length+100h+200h

;освободить всю память после конца программы и стека

mov ah,4Ah

stack\_shift = program\_length + 100h + 200h

mov bx,stack\_shift shr 4 + 1 ; размер в параграфах + 1

int 21h

; заполнить поля ЕРВ, содержащие сегментные адреса

mov ax,cs

mov word ptr EPB+4,ax

mov word ptr EPB+8,ax

mov word ptr EPB+0Ch,ax

; вызвать программу

mov ax,4B00h

mov dx,offset program\_path

mov bx,offset EPB

int 21h

; сегмент командной строки

; сегмент первого FCB

; сегмент второго FCB

; функция DOS 4Bh

; путь к файлу

; блок EPB

; запустить программу

; выход из программы

int 20h ; ret нельзя - стек перемещен

; данные программы

program\_path db "С:\test.exe",0 ; имя файла (ASCIZ-строка)

EPB dw 0000 ; текущее окружение

dw offset commandline,0 ; адрес командной строки

dw 005Ch,0,006Ch,0

commandline db 125

db " /?"

command\_text db 122 dup (?)

program\_length equ $-start

end start

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. Пример ассемблер- ного кода оверлейного модуля:

; OVERLAY1.ASM

DSEG SEGMENT ; сегмент данных оверлея

... DSEG ENDS

CSEG SEGMENT PARA PUBLIC 'CODE' ; сегмент кода оверлея

OVERLAY PROC FAR ; процедура оверлея

ASSUME CS:CSEG,DS:DSEG

PUSH DS ; сохранение DS вызывающей программы

MOV AX,DSEG ; установка DS на данные оверлея

MOV DS,AX

...

POP DS

RET

OVERLAY ENDP

CSEG ENDS

END

; тело оверлея

; восстановление DS при завершении

Пример загрузки скомпилированного оверлейного модуля в формате exe приведен ниже.

ZSEG SEGMENT ; фиктивный сегмент для расчета размера

ZSEG ENDS

...

; в сегменте данных основной программы

OVERLAY\_SEG DW ?

OVERLAY\_OFFSET DW ?

CODE\_SEG DW ?

PATH DB 'A:\OVERLAY1.EXE' ; путь к файлу оверлея

0BLOCK DD 0 ; 4-х байтный блок параметров

...

; в сегменте кода основной программы

; освобождаем память

MOV CODE\_SEG,CS

MOV AX,ES

MOV BX,ZSEG

SUB BX,AX

MOV AH,4AH

INT 21H

JC SETBLK\_ERROR

; сегмент кода

; сегмент PSP

; сегмент конца программы

; размер памяти программы

; изменение размера блока памяти

; проверка ошибки

; отводим память для оверлея

; сегмент оверлея для нулевого смещения

; смещение оверлея в сегменте кода

; сегмент оверлея = сегменту кода

MOV BX,100H

MOV AH,48H

INT 21H

JC ALLOC\_ERROR

MOV OVERLAY\_SEG,AX ; сохранение сегмента оверлея

; размер блока = 1000H байт

; выделение блока памяти

; AX:0000 указывает на блок памяти

; проверка ошибки

; вычисление смещения оверлея в сегменте кода

MOV AX,CODE\_SEG ; сегмент оверлея

MOV BX,OVERLAY\_SEG ; сегмент кода

SUB BX,AX ; BX = смещение в параграфах

MOV CL,4

SHL BX,CL ; сдвиг влево на 4 бита

MOV OVERLAY\_OFFSET,BX ; смещение в байтах в сегменте кода

; загрузка оверлея

MOV AX,SEG BLOCK

MOV ES,AX

MOV BX,OFFSET BLOCK

MOV AX,OVERLAY\_SEG

MOV [BX],AX

MOV [BX]+2,AX

; ES:BX указывает на блок параметров

; сегмент оверлея

; фактор привязки (сегмент оверлея)

LEA DX,PATH

MOV AH,4BH

MOV AL,3

INT 21H

; DS:DX = путь к файлу оверлея

; номер функции загрузки программы

; код загрузки оверлея

; выполнение загрузки оверлея

; проверка ошибки

JC LOAD\_ERROR

...

; вызов оверлея как далекой процедуры

CALL DWORD PTR OVERLAY\_OFFSET

...

Код программы:

.model tiny

.code

org 80h

length\_cmd db ?

command\_line db ?

org 100h

start:

cld ;clear DF

mov bp, sp

mov cl, length\_cmd

cmp cl, 1 ;check the length of the line

jle exit

;mov cx, -1

mov di, offset command\_line ;the beginning of the command line

;

read\_parameters:

lodsb ;load the symbol from the line

cmp al, 0Dh ;check, if the end of the line is reached

je no\_more\_params

cmp al, 20h ;if the space is reached

jne read\_parameters

dec si

mov byte ptr [si], 0

mov di, si

inc di

jmp second\_param

no\_more\_params:

dec si

mov byte ptr [si], 0

second\_param:

mov al, ' '

repe scasb

dec di

inc word ptr arg\_num

mov si, di ;the current offset of the command line

mov di, offset number

continue\_read:

cmp [si], 0Dh ;check, if the end of the line is reached

je prelast\_param

cmp [si], 20h

je prelast\_param

movsb

jmp continue\_read

prelast\_param:

mov byte ptr [si], 0

mov si, offset number

string\_to\_int:

call stri

mov sp, length\_prog+100h+200h ;shift the stack by 200h after the end of the program segment

mov ah, 4Ah ;clear the program memory

offset\_stack = length\_prog+100h+200h

mov bx, offset\_stack shr 4 + 1

int 21h

mov ax, cs

mov word ptr EPB+4, ax ;command line segment

mov word ptr EPB+8, ax ;the first FCB segment

mov word ptr EPB+0Ch, ax ;the second FCB segment

mov cx, N

for:

mov ax, 4B00h ;load and run the program

mov dx, offset command\_line+1 ;the beginning of the command line, path to the file

mov bx, offset EPB

int 21h

loop for

exit:

int 20h ;can't use ret, because the stack is shifted

EPB dw 0000 ;the current surroundings

dw offset commandline, 0 ;the adress of the command line

dw 005Ch, 0, 006Ch, 0 ;the addresses of FCB

commandline db 125 ;the length of the command line

db " /?" ;command line (3)

command\_text db 122 dup(?) ;(122)

number db 80 dup(0)

arg\_num dw 0

N dw 0

ten dw 10

length\_prog equ $-start

stri proc near

xor dx, dx

loop:

xor ax, ax

lodsb

test al, al ;if al && al - 0

jz done

cmp al, '9' ;if symbol isn't a digit

jnbe loop

cmp al, '0'

jb loop

sub ax, '0'

push ax

mov ax, dx

mov dx, 10

mul dx

mov dx, ax

pop ax

add dx, ax ;converted number is stored in cx

jmp loop

done:

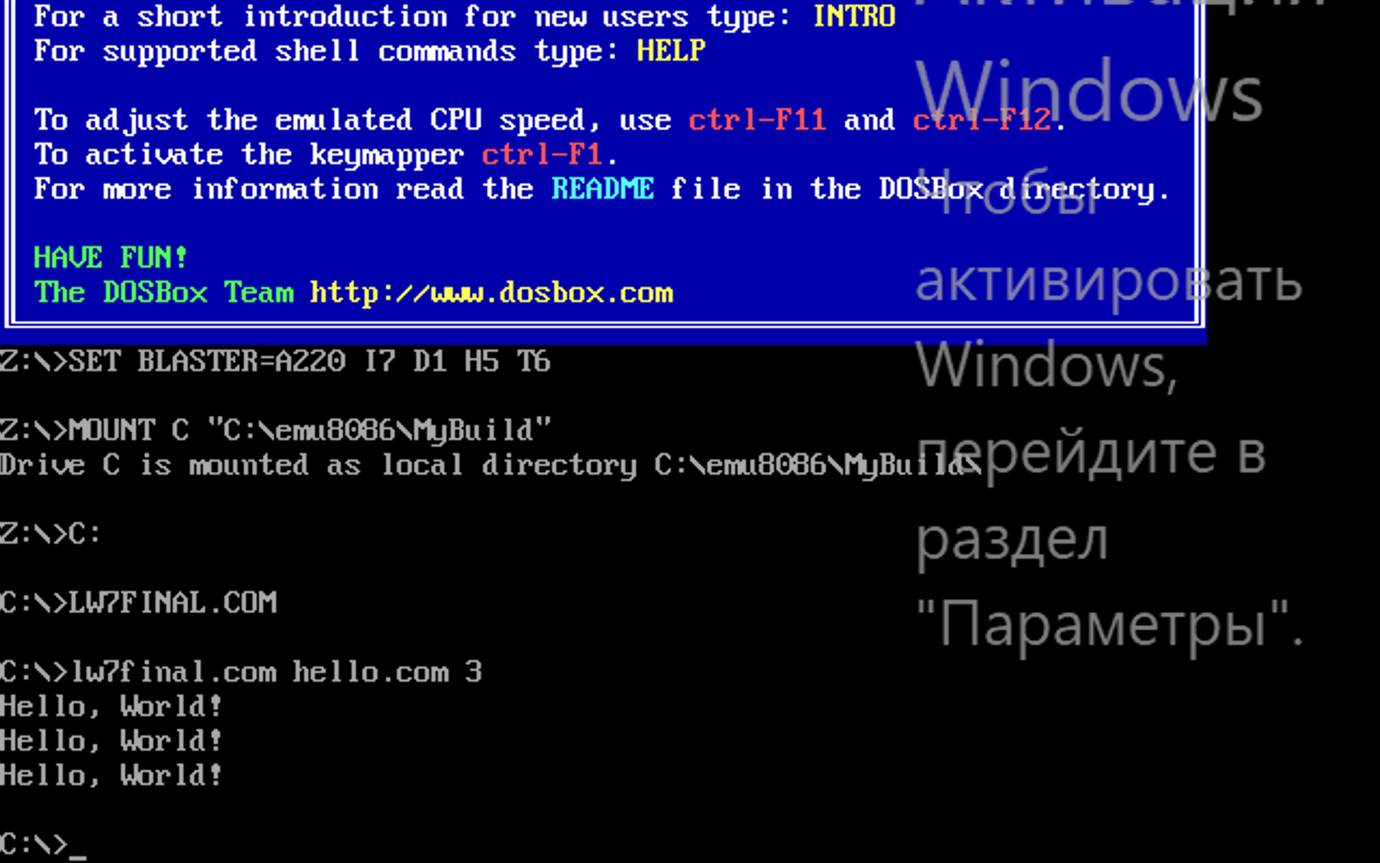
mov ax, dx

mov N, ax

stri endp

end start

Примеры работы программы:



Вывод:

В ходе лабораторной работы было написано приложение, выполняющее запуск другой программы, имя которой задается через командную строку, N раз. Я ознакомилась с техникой загрузки и выполнения программ, а также рассмотрела методы работы с памятью.