МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Операционные системы»

Тема: Построение модуля оверлейной структуры

Студент гр. 7381 _____ Трушников А.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследовать возможности построения загрузочного модуля оверлейной структуры. Исследуется структура оверлейного сегмента и способ загрузки и выполнения оверлейных сегментов. Для запуска вызываемого оверлейного модуля используется функция 4B03h прерывания int 21h. Все загрузочные и оверлейные модули должны находиться в одном каталоге.

Описание функций.

Название	Описание
Write_message	Вывод сообщения на экран
Main	Основная функция
Get_Path	Функция нахождения пути до вызываемого файла (в bp - имя файла)
Call_OVL	Функция вызова оверлейной программы при помощи 4B03h прерывания 21h

Описание структур данных.

Название	Описание
error_4Ah	Сообщение об ошибке в 4Ah
error4Ah_7	Сообщение об ошибке с кодом 7
error4Ah_8	Сообщение об ошибке с кодом 8
error4Ah_9	Сообщение об ошибке с кодом 9
error_4B03h	Сообщение об ошибке в 4B03h
error_4B03h_1	Сообщение об ошибке с кодом 1
error_4B03h_2	Сообщение об ошибке с кодом 2
error_4B03h_3	Сообщение об ошибке с кодом 3
error_4B03h_4	Сообщение об ошибке с кодом 4
error_4B03h_5	Сообщение об ошибке с кодом 5
error_4B03h_8	Сообщение об ошибке с кодом 8
error_4B03h_10	Сообщение об ошибке с кодом 10
memory_error_message	Сообщение об ошибки по причине слишком нехватки памяти для загрузки файла
error_4Eh	Сообщение об ошибке в 4Eh
error_4Eh_2	Сообщение об ошибке с кодом 2
error_4Eh_3	Сообщение об ошибке с кодом 3
adr	Адрес, по которому производится вызов оверлея

DTA	Буфер DTA
Keep_PSP	Переменная для хранения PSP
OVL_address	Блок параметров
DTA_paraghr	Путь к оверлею
name1	Имя 1-ого файла
name2	Имя 2-ого файла

Выполнение работы.

Шаг 1.

```
C:\>LAB7.EXE

There is the first overlay.
Its segment adress: 03B3

There is the second overlay.
Its segment adress: 03B3

C:\>
```

Программа и оверлеи в текущем каталоге.

Шаг 2.

```
C:\>\tt\lab7

There is the first overlay.
Its segment adress: 03B3

There is the second overlay.
Its segment adress: 03B3

C:\>
```

Программа вызывается не из текущего каталога. Программа выполняется нормально.

Шаг 3.

```
Z:\>c:

C:\>\tt\lab7
Overlay size cant be calculate! (func 4Eh error)
File not found (code 2)
Overlay program was not loaded! (func 4B03h error)
File not found (code 2)

There is the second overlay.
Its segment adress: 03B3

C:\>
```

Первый оверлей отсутствует в каталоге. Программа выдаёт сообщение об ошибке.

Ответы на контрольные вопросы.

1. Как должна быть устроена программа, если в качестве оверлейного сегмента использовать .COM модули?

.COM-файлы имеют структуру, согласно которой непосредственно код располагается с адреса 100h. Пространство от начала файла зарезервировано по PSP. Так как вызываемой оверлейной структуре функцией 4B03h прерывания 21h не передаётся управление и при этом не создаётся PSP, мы будем «обмануты» функцией 4Eh прерывания 21h, которая в данному случае неправильно определит размер оверлея. Нам придётся вручную корректировать полученный размер оверлея.

Выводы.

В ходе данной лабораторной работы я научился организовывать программы, имеющие оверлейную структуру, а также вызывать такие программы из управляющей программы при помощи функции 4B03h прерывания int 21h.

ПРИЛОЖЕНИЕ А

LAB7.ASM

```
ASTACK          SEGMENT  STACK
    DW 256 DUP(?)
ASTACK          ENDS

DATA            SEGMENT
        ERROR_4AH    DB 'MEMORY CAN NOT BE FREED! (FUNC 4AH ERROR)', 0DH, 0AH, '$'
          ;ПРИЧИНА НЕУДАЧНОГО ВЫПОЛНЕНИЯ ФУНКЦИИ 4AH
      ERROR4AH_7  DB 'CONTROL UNIT MEMORY DESTROYED (CODE 7)', 0DH, 0AH, '$'
        ;РАЗРУШЕН УПРАВЛЯЮЩИЙ БЛОК ПАМЯТИ (КОД 7)
      ERROR4AH_8        DB 'NOT ENOUGHT MEMORY TO PERFORM THE FUNCTION (CODE 8)',
0DH, 0AH, '$'   ;НЕДОСТАТОЧНО ПАМЯТИ ДЛЯ ВЫПОЛНЕНИЯ ФУНКЦИИ (КОД 8)
        ERROR4AH_9        DB 'INVALID ADRESS OF THE MEMORY BLOCK (CODE 9)', 0DH, 0AH,
'$'                     ;НЕВЕРНЫЙ АДРЕС БЛОКА ПАМЯТИ (КОД 9)

        ERROR_4B03H   DB 'OVERLAY PROGRAM WAS NOT LOADED! (FUNC 4B03H
ERROR)', 0DH, 0AH, '$'          ;ПРИЧИНА НЕУДАЧНОГО ЗАВЕРШЕНИЯ ФУНКЦИИ 4B03H

        ERROR_4B03H_1       DB 'INCORRECT NUMBER OF THE FUNCTION (CODE 1)', 0DH,
0AH, '$'                ;НЕВЕРНЫЙ НОМЕР ФУНКЦИИ (КОД 1)
        ERROR_4B03H_2   DB 'FILE NOT FOUND (CODE 2)', 0DH, 0AH, '$'
                        ;ФАЙЛ НЕ НАЙДЕН (КОД 2)
        ERROR_4B03H_3   DB 'WAY NOT FOUND (CODE 3)', 0DH, 0AH, '$'
                        ;МАРШРУТ НЕ НАЙДЕН (КОД 3)
        ERROR_4B03H_4   DB 'TOO MANY OPENED FILES (CODE 4)', 0DH, 0AH, '$'
                      ;СЛИШКОМ МНОГО ОТКРЫТЫХ ФАЙЛОВ (КОД 4)
        ERROR_4B03H_5   DB 'NO ACSESS (CODE 5)', 0DH, 0AH, '$'
                        ;НЕТ ДОСТУПА (КОД 5)
        ERROR_4B03H_8   DB 'NOT ENOUGHT MEMORY(CODE 8)', 0DH, 0AH, '$'
                        ;МАЛО ПАМЯТИ (КОД 8)
        ERROR_4B03H_10  DB 'INCORRECT ENVIRONMENT (CODE 10)', 0DH, 0AH, '$'
                        ;НЕПРАВИЛЬНАЯ СРЕДА(КОД 10)

        ERROR_4EH       DB 'OVERLAY SIZE CANT BE CALCULATE! (FUNC 4EH ERROR)',
0DH, 0AH, '$'   ;ПРИЧИНА НЕУДАЧНОГО ЗАВЕРШЕНИЯ ФУНКЦИИ 4EH

        ERROR_4EH_2     DB 'FILE NOT FOUND (CODE 2)', 0DH, 0AH, '$'
                      ;ФАЙЛ НЕ НАЙДЕН (КОД 2)
        ERROR_4EH_3     DB 'WAY NOT FOUND (CODE 3)', 0DH, 0AH, '$'
                        ;ПУТЬ НЕ НАЙДЕН (КОД 3)

        MEMORY_ERROR_MESSAGE  DB 'ERROR: TOO MANY BIG FILE', 0DH, 0AH, '$'
                        ;ОШИБКА ВОЗНИКАЕТ, КОГДА ФАЙЛ СЛИШКОМ ВЕЛИК
ДЛЯ ЗАГРУЗКИ

        ADR             DD      0
```

```
        DTA             DB      43 DUP (0), '$'
        KEEP_PSP        DW      0
        OVL_ADDRESS DW  0
        DTA_PARAGHR DB  256     DUP (0), '$'
        ; ИМЕНА ВЫЗЫВАЕМЫХ ФАЙЛОВ
        NAME1                   DB      'OVL_1',0
        NAME2                   DB      'OVL_2',0
DATA            ENDS

CODE    SEGMENT
        .386
   ASSUME CS:CODE, DS:DATA, SS:ASTACK

; ФУНКЦИЯ ВЫВОДА СООБЩЕНИЯ НА ЭКРАН
WRITE_MESSAGE       PROC
        PUSH AX
        MOV AH, 09H
        INT 21H
        POP AX
        RET
WRITE_MESSAGE       ENDP

; ФУНКЦИЯ, ОПРЕДЕЛЯЮЩАЯ РАЗМЕР ОВЕРЛЕЯ ПРИ ПОМОЩИ ФУНКЦИИ 4EH ПРЕРЫВАНИЯ 21H
MEMORYSIZE  PROC
        PUSH ES
        PUSH BX
        PUSH SI

        PUSH DS
        PUSH DX
        MOV DX, SEG DTA
        MOV DS, DX
        MOV DX, OFFSET DTA          ;В DS:DX - АДРЕС ДЛЯ DTA
        MOV AX, 1A00H               ;ФУНКЦИЯ УСТАНОВКИ АДРЕСА ДЛЯ DTA
        INT 21H
        POP DX
        POP DS

        PUSH DS
        PUSH DX
        MOV CX, 0                   ;ЗНАЧЕНИЕ БАЙТА АТРИБУТОВ ДЛЯ ФАЙЛА - 0
        MOV DX, SEG DTA_PARAGHR
        MOV DS, DX
        MOV DX, OFFSET DTA_PARAGHR  ;В DS:DX УКАЗАТЕЛЬ НА ПУТЬ К ФАЙЛУ
        MOV AX, 4E00H
        INT 21H
        POP DX
```

```
        POP DS

        JNC READ_SIZE

        LEA DX, ERROR_4EH
        CALL WRITE_MESSAGE

        CMP AX, 2
        JE ERROR4EH_2_LABEL

        CMP AX, 3
        JE ERROR4EH_3_LABEL

    ERROR4EH_2_LABEL:
        LEA DX, ERROR_4EH_2
        CALL WRITE_MESSAGE
        JMP END_FUNCTION

    ERROR4EH_3_LABEL:
        LEA DX, ERROR_4EH_3
        CALL WRITE_MESSAGE
        JMP END_FUNCTION

    READ_SIZE:
        PUSH ES
        PUSH BX
        PUSH SI
        MOV SI, OFFSET DTA
        ADD SI, 1CH          ;В БУФЕРЕ СО СМЕЩЕНИЕМ 1CH - СТАРШЕЕ СЛОВО
РАЗМЕРА ФАЙЛА
        MOV BX, [SI]  ;ЧИТАЕМ СТАРШЕЕ СЛОВО РАЗМЕРА ФАЙЛА
        CMP BX, 000FH
        JLE NO_MEMORY_ERROR
        JMP MEMORY_ERROR

    NO_MEMORY_ERROR:
        SUB SI, 2       ;ВЫБИРАЕМ СМЕЩЕНИЕ НА 1AH
        MOV BX, [SI]  ;ЧИТАЕМ ОТТУДА МЛАДШЕЕ СЛОВО РАЗМЕРА ФАЙЛА
        PUSH CX
        MOV CL, 4
        SHR BX, CL ;ПЕРЕВОД В ПАРАГРАФЫ (В BX - МЛАДШЕЕ СЛОВО)
        POP CX
        MOV AX, [SI+2] ;СНОВА ЧИТАЕМ СТАРШЕЕ СЛОВО
        PUSH CX
        MOV CL, 12
        SAL AX, CL    ;ПЕРЕВОДИМ В БАЙТЫ, А ЗАТЕМ В ПАРАГРАФЫ
        POP CX
        ADD BX, AX    ;ДОБАВЛЯЕМ К BX
```

```
        INC BX
        INC BX

        MOV AX, 4800H       ;ВЫЗЫВАЕМ ФУНКЦИЮ ВЫДЕЛЕНИЯ ПАМЯТИ
        INT 21H             ;С АРГУМЕНТОМ - КОЛИЧЕСТВОМ ПАРАГРАФОВ - В
BX
        MOV OVL_ADDRESS, AX ;СОХРАНЯЕМ ПОЛУЧЕННОЕ ЗНАЧЕНИЕ
        POP SI
        POP BX
        POP ES
        JMP END_FUNCTION

MEMORY_ERROR:
        LEA DX, MEMORY_ERROR_MESSAGE
        CALL WRITE_MESSAGE
        POP SI
        POP BX
        POP ES

END_FUNCTION:
        POP SI
        POP BX
        POP ES
        RET
MEMORYSIZE  ENDP


; ФУНКЦИЯ НАХОЖДЕНИЯ ПУТИ ДО ВЫЗЫВАЕМОГО ФАЙЛА (В BP - ИМЯ ФАЙЛА)
GET_PATH    PROC
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        PUSH SI
        PUSH DI
        PUSH ES

        MOV ES, KEEP_PSP
        MOV AX, ES:[2CH]
        MOV ES, AX
        MOV BX, 0
        MOV CX, 2

ENV_LOOP_PATH_LOCATE:
        INC CX
        MOV AL, ES:[BX]
        INC BX
        CMP AL, 0
```

```
        JZ      PRE_END_ENV_PATH_LOCATE
        LOOP ENV_LOOP_PATH_LOCATE

PRE_END_ENV_PATH_LOCATE:
        CMP BYTE PTR ES:[BX], 0
        JNZ ENV_LOOP_PATH_LOCATE
        ADD BX, 3
        LEA SI, DTA_PARAGHR

PATH_LOOP_PATH_LOCATE:
        MOV AL, ES:[BX]
        MOV [SI], AL
        INC SI
        INC BX
        CMP AL, 0
        JZ      END_PATH_LOOP_PATH_LOCATE_M
        JMP PATH_LOOP_PATH_LOCATE

END_PATH_LOOP_PATH_LOCATE_M:
        SUB SI, 9
        MOV DI, BP

REPLACE_LOOP_PATH_LOCATE:
        MOV AH, [DI]
        MOV [SI], AH
        CMP AH, 0
        JZ      END_REPLACE_LOOP_PATH_LOCATE
        INC DI
        INC SI
        JMP REPLACE_LOOP_PATH_LOCATE

END_REPLACE_LOOP_PATH_LOCATE:
        POP ES
        POP DI
        POP SI
        POP DX
        POP CX
        POP BX
        POP AX
        RET
GET_PATH    ENDP

; ФУНКЦИЯ ВЫЗОВА ОВЕРЛЕЙНОЙ ПРОГРАММЫ ПРИ ПОМОЩИ 4B03H ПРЕРЫВАНИЯ
21H
CALL_OVL    PROC
        PUSH AX
        PUSH BX
        PUSH CX
```

```
        PUSH DX
        PUSH BP

        MOV BX, SEG OVL_ADDRESS
        MOV ES, BX
        MOV BX, OFFSET OVL_ADDRESS      ;В ES:BX - УКАЗАТЕЛЬ НА БЛОК ПАРАМЕТРОВ

        MOV DX, SEG DTA_PARAGHR
        MOV DS, DX
        MOV DX, OFFSET DTA_PARAGHR      ;В DS:DX - УКАЗАТЕЛЬ НА ПУТЬ К ОВЕРЛЕЮ

        PUSH SS
        PUSH SP

        MOV AX, 4B03H           ;ВЫЗЫВАЕМ ФУНКЦИЮ
        INT 21H
        JNC NO_ERROR1

        LEA DX, ERROR_4B03H
        CALL WRITE_MESSAGE

        CMP AX, 1
        JE ERROR_4B03H_1_LABEL

        CMP AX, 2
        JE ERROR_4B03H_2_LABEL

        CMP AX, 3
        JE ERROR_4B03H_3_LABEL

        CMP AX, 4
        JE ERROR_4B03H_4_LABEL

        CMP AX, 5
        JE ERROR_4B03H_5_LABEL

        CMP AX, 8
        JE ERROR_4B03H_8_LABEL

        CMP AX, 10
        JE ERROR_4B03H_10_LABEL

        JMP ERROR1

ERROR_4B03H_1_LABEL:
        LEA DX, ERROR_4B03H_1
        CALL WRITE_MESSAGE
        JMP ERROR1
```

```
ERROR_4B03H_2_LABEL:
        LEA DX, ERROR_4B03H_2
        CALL WRITE_MESSAGE
        JMP ERROR1

ERROR_4B03H_3_LABEL:
        LEA DX, ERROR_4B03H_3
        CALL WRITE_MESSAGE
        JMP ERROR1

ERROR_4B03H_4_LABEL:
        LEA DX, ERROR_4B03H_4
        CALL WRITE_MESSAGE
        JMP ERROR1

ERROR_4B03H_5_LABEL:
        LEA DX, ERROR_4B03H_5
        CALL WRITE_MESSAGE
        JMP ERROR1

ERROR_4B03H_8_LABEL:
        LEA DX, ERROR_4B03H_8
        CALL WRITE_MESSAGE
        JMP ERROR1

ERROR_4B03H_10_LABEL:
        LEA DX, ERROR_4B03H_10
        CALL WRITE_MESSAGE
        JMP ERROR1

NO_ERROR1:
        MOV AX, SEG DATA
        MOV DS, AX    ;ВОССТАНАВЛИВАЕМ DS
        MOV AX, OVL_ADDRESS
        MOV WORD PTR ADR+2, AX
        CALL ADR
        MOV AX, OVL_ADDRESS
        MOV ES, AX
        MOV AX, 4900H
        INT 21H
        MOV AX, SEG DATA
        MOV DS, AX

ERROR1:
        POP SP
        POP SS
        MOV ES, KEEP_PSP
```

```
        POP BP
        POP DX
        POP CX
        POP BX
        POP AX
        RET
CALL_OVL    ENDP

; ГЛАВНАЯ ФУНКЦИЯ
MAIN    PROC
        MOV AX, SEG DATA
        MOV DS, AX
        MOV KEEP_PSP, ES

;ПОДГОТОВКА МЕСТА В ПАМЯТИ
        MOV AX, ALL_MEMORY      ;ВСЯ ПАМЯТЬ, ВЫДЕЛЕННАЯ ПРОГРАММЕ
        MOV BX, ES              ;ИСПОЛЬЗУЕМАЯ ПАМЯТЬ
        SUB AX, BX              ;ВЫЧИСЛЯЕМ ОСТАТОК
        MOV CX, 0004H
        SHR AX, CL              ;ПЕРЕВОДИМ В ПАРАГРАФЫ
        MOV BX, AX              ;УКАЗЫВАЕМ ВХОДНОЙ ПАРАМЕТР ДЛЯ ФУНКЦИИ 4AH

        MOV AX, 4A00H
        INT 21H                 ;ВЫПОЛНЯЕМ ФУНКЦИЮ (В AX ЗАНЕСЁТСЯ КОД ОШИБКИ ЕСЛИ ОНА БУДЕТ)

;ОБРАБОТКА ОШИБОК ФУНКЦИИ 4AH
        JNC NO_ERROR_4AH  ;ЕСЛИ ФЛАГ CF=0 - ОШИБОК НЕ БЫЛО, ДВИЖЕМСЯ ДАЛЕЕ

        LEA DX, ERROR_4AH ;ВЫВОДИМ СООБЩЕНИЕ О ТОМ, ЧТО ПРОИЗОШЛА ОШИБКА В ФУНКЦИИ 4AH
        CALL WRITE_MESSAGE

        CMP AX, 7               ;ОШИБКА С КОДОМ 7
        JE ERROR4AH_7_LABEL ;СООТВЕТСТВУЮЩАЯ ОБРАБОТКА

        CMP AX, 8               ;ОШИБКА С КОДОМ 8
        JE ERROR4AH_8_LABEL ;СООТВЕТСТВУЮЩАЯ ОБРАБОТКА

        CMP AX, 9               ;ОШИБКА С КОДОМ 9
        JE ERROR4AH_8_LABEL ;СООТВЕТСТВУЮЩАЯ ОБРАБОТКА

ERROR4AH_7_LABEL:
        LEA DX, ERROR4AH_7
        CALL WRITE_MESSAGE
        JMP ERROR_FINISH
```

```
ERROR4AH_8_LABEL:
        LEA DX, ERROR4AH_8
        CALL WRITE_MESSAGE
        JMP ERROR_FINISH

ERROR4AH_9_LABEL:
        LEA DX, ERROR4AH_9
        CALL WRITE_MESSAGE
        JMP ERROR_FINISH

NO_ERROR_4AH:
;ОСВОБОЖДЕНИЕ МЕСТА, НАХОЖДЕНИЕ ПУТИ И ЗАПУСК 1 ОВЕРЛЕЯ
        LEA BP, NAME1
        CALL GET_PATH
        CALL MEMORYSIZE
        CALL CALL_OVL

;ОСВОБОЖДЕНИЕ МЕСТА, НАХОЖДЕНИЕ ПУТИ И ЗАПУСК 2 ОВЕРЛЕЯ
        LEA BP, NAME2
        CALL GET_PATH
        CALL MEMORYSIZE
        CALL CALL_OVL

ERROR_FINISH:
        MOV AH, 4CH             ;ЗАВЕРШЕНИЕ ПО ФУНКЦИИ 4C
        INT 21H
MAIN    ENDP
CODE                    ENDS

ALL_MEMORY              SEGMENT         ;ПУСТОЙ СЕГМЕНТ В КОНЦЕ
ALL_MEMORY  ENDS ;ДЛЯ ОПРЕДЕЛЕНИЯ ПАМЯТИ, КОТОРАЯ НЕ ИСПОЛЬЗУЕМСЯ В CS

                END MAIN
```

OVL_1.ASM

```
OVL1 SEGMENT
        ASSUME CS:OVL1, DS:NOTHING, SS:NOTHING, ES:NOTHING

MAIN PROC FAR
        PUSH DS
        PUSH AX
        PUSH DI
        PUSH DX
        PUSH BX

        MOV DS, AX
        LEA DX, CS:MESSAGE          ;ВЫВОДИМ СООБЩЕНИЕ О ТОМ, ЧТО
        CALL WRITE_MESSAGE          ;БЫЛ ВЫЗВАН ПЕРВЫЙ ОВЕРЛЕЙ

        LEA BX, CS:ADRESS
        ADD BX, 23                  ;УСТАНАВЛИВАЕМ КУРСОР В НУЖНОЕ МЕСТО
        MOV DI, BX                  ;В DI - АДРЕСС ПОСЛЕДНЕГО СИМВОЛА
        MOV AX, CS                  ;В AX - ЧИСЛО, НЕОБХОДИМОЕ
ПЕРЕКОНВЕРТИРОВАТЬ
        CALL WRD_TO_HEX

        LEA DX, CS:ADRESS   ;ВЫВОДИМ СООБЩЕНИЕ О ТОМ, ЧТО
        CALL WRITE_MESSAGE          ;БЫЛ ВЫЗВАН ВТОРОЙ ОВЕРЛЕЙ

        POP BX
        POP DX
        POP DI
        POP AX
        POP DS
        RETF
MAIN ENDP

MESSAGE     DB      10, 13, 'THERE IS THE FIRST OVERLAY.', 10, 13, '$'
ADRESS              DB      'ITS SEGMENT ADRESS:    ', 10, 13, '$'

; ФУНКЦИЯ ВЫВОДА СООБЩЕНИЯ НА ЭКРАН
WRITE_MESSAGE       PROC
        PUSH AX
        MOV AH, 09H
        INT 21H
        POP AX
        RET
WRITE_MESSAGE       ENDP

TETR_TO_HEX PROC NEAR
        AND AL,0FH
```

```
        CMP AL,09
        JBE NEXT
        ADD AL,07
NEXT:
        ADD AL,30H
        RET
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC NEAR
;БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА ШЕСТНАДЦАТЕРИЧНЫХ СИМВОЛА В AX
        PUSH CX
        MOV AH,AL
        CALL TETR_TO_HEX
        XCHG AL,AH
        MOV CL,4
        SHR AL,CL
        CALL TETR_TO_HEX ; В AL - СТАРШИЙ БАЙТ
        POP CX ;В AH - МЛАДШИЙ
        RET
BYTE_TO_HEX ENDP

WRD_TO_HEX PROC NEAR
;ПЕРЕВОД В ШЕСТНАДЦАТЕРИЧНУЮ СИСТЕМУ СЧИСЛЕНИЯ ЧИСЛА В AX
; В AX - НОМЕР, В DI - ССЫЛКА НА ПОСЛЕДНИЙ СИМВОЛ
        PUSH BX
        MOV BH,AH
        CALL BYTE_TO_HEX
        MOV [DI],AH
        DEC DI
        MOV [DI],AL
        DEC DI
        MOV AL,BH
        CALL BYTE_TO_HEX
        MOV [DI],AH
        DEC DI
        MOV [DI],AL
        POP BX
        RET
WRD_TO_HEX ENDP

OVL1 ENDS
END MAIN
```

OVL_2.ASM

```
OVL1 SEGMENT
        ASSUME CS:OVL1, DS:NOTHING, SS:NOTHING, ES:NOTHING

MAIN PROC FAR
        PUSH DS
        PUSH AX
        PUSH DI
        PUSH DX
        PUSH BX

        MOV DS, AX
        LEA DX, CS:MESSAGE          ;ВЫВОДИМ СООБЩЕНИЕ О ТОМ, ЧТО
        CALL WRITE_MESSAGE          ;БЫЛ ВЫЗВАН ПЕРВЫЙ ОВЕРЛЕЙ

        LEA BX, CS:ADRESS
        ADD BX, 23                  ;УСТАНАВЛИВАЕМ КУРСОР В НУЖНОЕ МЕСТО
        MOV DI, BX                  ;В DI - АДРЕСС ПОСЛЕДНЕГО СИМВОЛА
        MOV AX, CS                  ;В AX - ЧИСЛО, НЕОБХОДИМОЕ
ПЕРЕКОНВЕРТИРОВАТЬ
        CALL WRD_TO_HEX

        LEA DX, CS:ADRESS   ;ВЫВОДИМ СООБЩЕНИЕ О ТОМ, ЧТО
        CALL WRITE_MESSAGE          ;БЫЛ ВЫЗВАН ВТОРОЙ ОВЕРЛЕЙ

        POP BX
        POP DX
        POP DI
        POP AX
        POP DS
        RETF
MAIN ENDP

MESSAGE     DB      10, 13, 'THERE IS THE SECOND OVERLAY.', 10, 13, '$'
ADRESS              DB      'ITS SEGMENT ADRESS:    ', 10, 13, '$'

; ФУНКЦИЯ ВЫВОДА СООБЩЕНИЯ НА ЭКРАН
WRITE_MESSAGE   PROC
        PUSH AX
        MOV AH, 09H
        INT 21H
        POP AX
        RET
WRITE_MESSAGE   ENDP

TETR_TO_HEX PROC NEAR
        AND AL,0FH
        CMP AL,09
```

```
        JBE NEXT
        ADD AL,07
NEXT:
        ADD AL,30H
        RET
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC NEAR
;БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА ШЕСТНАДЦАТЕРИЧНЫХ СИМВОЛА В AX
        PUSH CX
        MOV AH,AL
        CALL TETR_TO_HEX
        XCHG AL,AH
        MOV CL,4
        SHR AL,CL
        CALL TETR_TO_HEX ; В AL - СТАРШИЙ БАЙТ
        POP CX ;В AH - МЛАДШИЙ
        RET
BYTE_TO_HEX ENDP

WRD_TO_HEX PROC NEAR
;ПЕРЕВОД В ШЕСТНАДЦАТЕРИЧНУЮ СИСТЕМУ СЧИСЛЕНИЯ ЧИСЛА В AX
; В AX - НОМЕР, В DI - ССЫЛКА НА ПОСЛЕДНИЙ СИМВОЛ
        PUSH BX
        MOV BH,AH
        CALL BYTE_TO_HEX
        MOV [DI],AH
        DEC DI
        MOV [DI],AL
        DEC DI
        MOV AL,BH
        CALL BYTE_TO_HEX
        MOV [DI],AH
        DEC DI
        MOV [DI],AL
        POP BX
        RET
WRD_TO_HEX ENDP

OVL1 ENDS
END MAIN
```