**Лабораторная работа №10**

**Цель работы**

Требуется скорректировать программу (и соответствующие файлы) так, чтобы экранная строка дампа содержала бы сначала 4-х позиционный шестнадцатеричный адрес первого байта строки, далее один пробел, после которого шестнадцатеричное представление 16-и байтов памяти. Далее следует еще один пробел, после которого следует символьное представление этих же 16-и байтов (между символами пробелов нет). При этом требуется скорректировать процедуру Write\_char так, чтобы вместо любого управляющего символа (код ASCII от 00h до 1Fh включительно) должен выводиться символ “.”. Пример экранной строки: А17F 41 42 43 44 45 46 47 00 20 1F 80 81 82 83 84 85 ABCDEFG. .АБВГДЕ

**Решение**

**1. Файл**Disp\_sec.asm**(основной)**

*; ====================================================*

*; Программа: Disp\_sec.asm*

*; Описание: Вывод дампа памяти с адресом, HEX и ASCII-представлением*

*; Формат: .COM (начинается с адреса 100h)*

*; ====================================================*

org 100h

jmp start

*; --------------------------*

*; Данные*

*; --------------------------*

Address dw 100h *; Начальный адрес области дампа*

Sector db 256 dup(?) *; Буфер для данных (256 байт)*

*; --------------------------*

*; Процедура: Вывод строки дампа*

*; Вход: DX = смещение строки (0, 16, 32, ...)*

*; --------------------------*

Disp\_line:

push bx

push ax

push cx

push dx

*; Вывод 4-значного адреса*

mov bx, dx

add bx, [Address] *; BX = адрес первого байта строки*

mov ax, bx

call Write\_word\_hex *; Выводим адрес (AX)*

mov dl, ' '

call Write\_char *; Пробел после адреса*

*; Вывод 16 байт в HEX*

mov cx, 16

mov si, dx *; SI = смещение в Sector*

.Hex\_loop:

mov al, [Sector + si]

call Write\_byte\_hex *; Вывод байта*

mov dl, ' '

call Write\_char *; Пробел между байтами*

inc si

loop .Hex\_loop

*; Вывод 16 байт в ASCII*

mov dl, ' '

call Write\_char *; Пробел перед ASCII*

mov cx, 16

mov si, dx *; Сброс SI*

.Ascii\_loop:

mov dl, [Sector + si]

call Write\_char *; Вывод символа (или '.')*

inc si

loop .Ascii\_loop

pop dx

pop cx

pop ax

pop bx

ret

*; --------------------------*

*; Процедура: Вывод символа (замена управляющих на '.')*

*; Вход: DL = символ*

*; --------------------------*

Write\_char:

push ax

cmp dl, 20h *; Проверка на управляющий символ*

jae .print *; Если >= 20h, выводим как есть*

mov dl, '.' *; Иначе заменяем на точку*

.print:

mov ah, 02h

int 21h

pop ax

ret

*; --------------------------*

*; Остальные процедуры (Write\_word\_hex, Write\_byte\_hex)*

*; ... (из предыдущих работ)*

*; --------------------------*

%include "Video\_io.asm" *; Подключаем вывод HEX*

*; --------------------------*

*; Точка входа*

*; --------------------------*

start:

*; Заполнение Sector тестовыми данными*

mov cx, 256

mov si, Sector

mov al, 0

.init\_loop:

mov [si], al

inc al

inc si

loop .init\_loop

*; Вывод дампа (первые 16 строк)*

mov dx, 0 *; Смещение первой строки*

mov cx, 16

.dump\_loop:

call Disp\_line

call New\_line

add dx, 16 *; Следующая строка*

loop .dump\_loop

mov ah, 4Ch *; Завершение программы*

int 21h

*; --------------------------*

*; Процедура: Перевод строки*

*; --------------------------*

New\_line:

push dx

mov dl, 0Dh

call Write\_char

mov dl, 0Ah

call Write\_char

pop dx

ret

**2. Файл**Video\_io.asm**(подключаемый)**

*; ====================================================*

*; Файл: Video\_io.asm*

*; Описание: Процедуры вывода HEX-чисел (из предыдущей работы)*

*; ====================================================*

*; Процедура: Вывод слова в HEX (AX)*

Write\_word\_hex:

push ax

push cx

push dx

mov cx, 4 *; 4 цифры*

.next\_digit:

rol ax, 4 *; Циклический сдвиг*

mov dl, al

and dl, 0Fh *; Изолируем 4 бита*

add dl, '0' *; В ASCII*

cmp dl, '9'

jbe .print

add dl, 7 *; Коррекция для A-F*

.print:

mov ah, 02h

int 21h

loop .next\_digit

pop dx

pop cx

pop ax

ret

*; Процедура: Вывод байта в HEX (AL)*

Write\_byte\_hex:

push ax

push dx

mov dl, al

shr dl, 4 *; Старшая тетрада*

call .print\_digit

mov dl, al

and dl, 0Fh *; Младшая тетрада*

call .print\_digit

pop dx

pop ax

ret

.print\_digit:

add dl, '0'

cmp dl, '9'

jbe .print

add dl, 7

.print:

mov ah, 02h

int 21h

ret

**3. Тестирование в DEBUG**

1. **Скомпилируйте**:

nasm Disp\_sec.asm -o Disp\_sec.com

1. **Запустите в DEBUG**:

DEBUG Disp\_sec.com

1. **Проверьте вывод**:
   * Первая строка должна выглядеть так:

0100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ................

* + Управляющие символы (коды 00h-1Fh) заменяются на .

1. **Измените**Address:

E 102 FF 00 ; Меняем Address на 00FFh

G ; Запуск (адреса начнутся с 00FFh)

**Ключевые изменения**

1. **Формат вывода**:
   * Адрес (4 HEX-цифры) + 16 байт в HEX + 16 символов.
2. **Замена управляющих символов**:
   * В Write\_char добавлена проверка DL < 20h.
3. **Использование стека**:
   * push bx/pop bx для сохранения номера байта.

**Пример вывода**

01A0 41 42 43 00 1F 20 80 FF A1 A2 A3 A4 A5 A6 A7 A8 ABC.. .АБВГДЕЖЗ

* Управляющие символы 00h и 1Fh заменены на ..
* Символы 80h-FFh выводятся как русские буквы (CP866).