**Лабораторная работа №9**

**Цель работы:**

Требуется написать и отладить две различные процедуры вывода на экран шестнадцатеричного числа, содержащегося в одном слове (в двух байтах). Первая из этих процедур использует тот же алгоритм, что и приведенная выше процедура вывода десятичных чисел. Вторая процедура Write\_word\_hex гораздо короче. Она выводит шестнадцатеричное содержимое слова путем двух вызовов процедуры Write\_byte\_hex, полученной на предыдущей лабораторной работе. При первом вызове на экран выводится шестнадцатеричное содержимое старшего байта слова, а при втором – младшего байта. Данная процедура потребуется нам на последующих лабораторных работах для вывода на экран 4-х значных адресов памяти. При этом в отличие от первой процедуры каждый адрес будет отображен на экране в виде ровно четырех символов. Для обеих разрабатываемых процедур требуется, чтобы тестирующая программа и сама процедура находились в разных исходных файлах.

**Решение:**

**1. Процедура 1: Поразрядный вывод (аналогично десятичному)**

**Алгоритм:**

1. Извлекаем каждую шестнадцатеричную цифру (от старшей к младшей).
2. Конвертируем цифру в ASCII-символ (0-9 → 30h-39h, A-F → 41h-46h).
3. Выводим символ через DOS (int 21h, функция 02h).

**Код (**write\_word\_hex1.asm**)**

*; Процедура вывода слова в HEX (поразрядно)*

*; Вход: AX = число для вывода*

*; Выход: Число на экране*

Write\_word\_hex1:

push cx

push dx

mov cx, 4 *; 4 цифры в слове*

.next\_digit:

rol ax, 4 *; Циклический сдвиг влево на 4 бита*

mov dl, al

and dl, 0Fh *; Изолируем младшие 4 бита*

add dl, '0' *; Конвертируем в ASCII*

cmp dl, '9'

jbe .print *; Если 0-9, выводим*

add dl, 7 *; Коррекция для A-F*

.print:

mov ah, 02h *; Функция вывода символа*

int 21h

loop .next\_digit *; Повторяем для всех цифр*

pop dx

pop cx

ret

**2. Процедура 2: Вывод через два байта (с использованием**Write\_byte\_hex**)**

**Алгоритм:**

1. Выводим старший байт (AH) через Write\_byte\_hex.
2. Выводим младший байт (AL) через Write\_byte\_hex.

**Код (**write\_word\_hex2.asm**)**

*; Процедура вывода слова в HEX (через байты)*

*; Вход: AX = число для вывода*

*; Выход: Число на экране*

Write\_word\_hex2:

push ax

push bx

mov bl, ah *; Сохраняем старший байт*

call Write\_byte\_hex *; Выводим его*

mov al, bl *; Восстанавливаем AL*

call Write\_byte\_hex *; Выводим младший байт*

pop bx

pop ax

ret

*; Предполагается, что Write\_byte\_hex уже определена*

Write\_byte\_hex:

push ax

push cx

mov ah, 0

mov cl, 16

div cl *; AH = остаток (младшая цифра), AL = частное (старшая цифра)*

call .print\_digit *; Вывод старшей цифры*

mov al, ah

call .print\_digit *; Вывод младшей цифры*

pop cx

pop ax

ret

.print\_digit:

add al, '0'

cmp al, '9'

jbe .print

add al, 7

.print:

mov dl, al

mov ah, 02h

int 21h

ret

**3. Тестирующая программа (**main.asm**)**

org 100h *; .COM-формат*

*; Подключаем процедуры*

%include "write\_word\_hex1.asm"

%include "write\_word\_hex2.asm"

start:

mov ax, 0ABCDh *; Тестовое число*

call Write\_word\_hex1 *; Вывод через поразрядный метод*

call new\_line

mov ax, 0ABCDh

call Write\_word\_hex2 *; Вывод через байты*

call new\_line

mov ah, 4Ch *; Завершение программы*

int 21h

new\_line:

mov ah, 02h

mov dl, 0Dh

int 21h

mov dl, 0Ah

int 21h

ret

**4. Компиляция и запуск**

1. Сохраним файлы:
   * write\_word\_hex1.asm
   * write\_word\_hex2.asm
   * main.asm
2. Скомпилируем и соберём программу (например, в **TASM** или **NASM**):

nasm main.asm -o main.com

1. Запустим в DOS или эмуляторе (DOSBox):

main.com

**Результат:**

ABCD

ABCD