**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

# ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине “Архитектура вычислительных систем”

Студент Станиславчук С. М.

Группа АС-21-1

Руководитель Болдырихин О. В.

Ст. преподаватель

Липецк 2023

**Цель работы:**

Изучение основ устройства и принципов работы компьютера принстонской архитектуры.

**Задание кафедры: Вариант 27**

Написать на языке ассемблера программу, выполняющую преобразование числа в упакованный двоично-десятичный код.

При помощи отладчика прогнать программу покомандно и после выполнения каждой команды фиксировать состояние аккумулятора, указателя команд, других регистров, задействованных в программе, ячеек памяти данных.

**Написать в программу подпрограммы: ближнюю и дальнюю. В подпрограмме должен быть стек. Нужно, чтобы хотя бы один параметр некоторой подпрограммы выполнялся через стек.**

Результаты анализа работы программы оформить в виде таблицы. Последовательность строк в таблице должна соответствовать последовательности выполнения команд в период прогона программы, а не их последовательности в тексте программы. В строке, соответствующей данной команде, содержимое регистров и памяти должно быть таким, каким оно является после ее выполнения.

Проанализировать таблицу, выполнить необходимые сравнения, сделать выводы.

| № | Задача, выполняемая программой | Расположение исходных данных | Расположение результата |
|---|---|---|---|
| 27 | Преобразование числа в упакованный двоично-десятичный код | Дополнительный сегмент данных (по ES) | Сегмент данных (по DS) и сегмент команд |

## Ход работы:

## 1. Код программы

```
model small
data_in segment
    input db 83
data_in ends
data_out segment
    res1 db 0
data_out ends
stack segment
    dw 100 dup(0) ; Stack definition
stack ends
code segment
    res2 db 0
    assume DS:data_out, ES:data_in, CS:code, SS:stack

near_conversion proc
    mov ax, data_in
    mov es, ax
    mov ax, data_out
    mov ds, ax

    mov al, input
    xor ah, ah

    mov bl, 10
    div bl
    mov dl, al
    mov al, ah
    shl dl, 4
    or al, dl

    ; Push the value of ax onto the stack
    push ax

    ; Call far_conversion
    call far ptr far_conversion

    ret; Return from subroutine
near_conversion endp

far_conversion proc far

    ; Store the low nibble in res1
    mov res1, al

    ; Store the high nibble in res2
    mov res2, al

    retf 2; Return from far subroutine
far_conversion endp
```

```
start:
    mov ax, stack
    mov ss, ax ; Stack initialization

    call near_conversion

    ; Now res2 contains the final result

    mov ah, 4Ch
    int 21h

code ends
end start
```

## 2. Листинг программы

```
1     0034   B8B948        mov ax, stack          0037   ax 48B9
2     0037   8ED0          mov ss, ax             0039   sp FFFE, ss 48B9
3     0039   E8C5FF        call near_conversion   0001   -
4     0001   B8B748        mov ax, data_in        0004   ax 48B7
5     0004   8EC0          mov es, ax             0006   es 48B7
6     0006   B8B848        mov ax, data_out       0009   ax 48B8
7     0009   8ED8          mov ds, ax             000B   ds 48B8, es:0000 = 53
8     000B   26A00000      mov al, input          000F   ax 4853
9     000F   32E4          xor ah, ah             0011   ax 0053
10    0011   B30A          mov bl, 10             0013   bx F60A
11    0013   F6F3          div bl                 0015   ax 0308
12    0015   8AD0          mov dl, al             0017   dx B408
13    0017   8AC4          mov al, ah             0019   ax 0303
14    0019   D0E2          shl dl, 1              001B   dx B410
15    001B   D0E2          shl dl, 1              001D   dx B420
16    001D   D0E2          shl dl, 1              001F   dx B440
17    001F   D0E2          shl dl, 1              0021   dx B480
20    0021   0AC2          or al, dl              0023   ax 0383
21    0023   50            push ax                0024   sp FFFC
22    0024   0E            push cs                0025   sp FFFA
23    0025   E80200        call far_conversion    0028   sp FFFE
24    0028   90            nop                    0029   -
25    0029   C3            ret                    003C   sp 0000
26    003C   B44C          mov ah, 4Ch            003E   ax 4C83
27    003E   CD21          int 21h                -      -
```

### 3. Таблица состояния системы

Составим таблицу состояний системы после выполнения каждой команды (таблица 1)

Таблица 1 – Состояния системы после выполнения команд программы

| Номер команды | Адрес команды | Команда на машинном языке | Регистр команд | Команда на языке ассемблера | Указатель команд | Содержание изменившихся регистров и ячеек памяти |
|---|---|---|---|---|---|---|
| 1 | 0034 | B8B948 | B8B948 | mov ax, stack | 0037 | ax 48B9 |
| 2 | 0037 | 8ED0 | 8ED0 | mov ss, ax | 0039 | sp FFFE, ss 48B9 |
| 3 | 0039 | E8C5FF | E8C5FF | call near_conversion | 0001 | - |
| 4 | 0001 | B8B748 | B8B748 | mov ax, data_in | 0004 | ax 48B7 |
| 5 | 0004 | 8EC0 | 8EC0 | mov es, ax | 0006 | es 48B7 |
| 6 | 0006 | B8B848 | B8B848 | mov ax, data_out | 0009 | ax 48B8 |
| 7 | 0009 | 8ED8 | 8ED8 | mov ds, ax | 000B | ds 48B8, es:0000 = 53 |
| 8 | 000B | 26A00000 | 26A00000 | mov al, input | 000F | ax 4853 |
| 9 | 000F | 32E4 | 32E4 | xor ah, ah | 0011 | ax 0053 |
| 10 | 0011 | B30A | B30A | mov bl, 10 | 0013 | bx F60A |
| 11 | 0013 | F6F3 | F6F3 | div bl | 0015 | ax 0308 |
| 12 | 0015 | 8AD0 | 8AD0 | mov dl, al | 0017 | dx B408 |
| 13 | 0017 | 8AC4 | 8AC4 | mov al, ah | 0019 | ax 0303 |
| 14 | 0019 | D0E2 | D0E2 | shl dl, 1 | 001B | dx B410 |
| 15 | 001B | D0E2 | D0E2 | shl dl, 1 | 001D | dx B420 |
| 16 | 001D | D0E2 | D0E2 | shl dl, 1 | 001F | dx B440 |
| 17 | 001F | D0E2 | D0E2 | shl dl, 1 | 0021 | dx B480 |
| 20 | 0021 | 0AC2 | 0AC2 | or al, dl | 0023 | ax 0383 |
| 21 | 0023 | 50 | 50 | push ax | 0024 | sp FFFC |
| 22 | 0024 | 0E | 0E | push cs | 0025 | sp FFFA |
| 23 | 0025 | E80200 | E80200 | call far_conversion | 0028 | sp FFFE |
| 24 | 0028 | 90 | 90 | nop | 0029 | - |
| 25 | 0029 | C3 | C3 | ret | 003C | sp 0000 |
| 26 | 003C | B44C | B44C | mov ah, 4Ch | 003E | ax 4C83 |
| 27 | 003E | CD21 | CD21 | int 21h | - | - |

## 4. Проверка работы алгоритма на правильных числах

Упакованный двоично-десятичный код (Packed Binary Coded Decimal, PBCD) - это способ представления десятичных чисел в формате, где каждая десятичная цифра представлена в виде 4-битного двоичного числа. В упакованном PBCD каждая десятичная цифра (0-9) кодируется с использованием 4 битов, и эти коды объединяются вместе, чтобы представить десятичное число.

На вход программе подается число 83. Далее в ближней подпрограмме разбивается это число на составные цифры (8 и 3) и заносит их в сегмент ES. После разбиения происходит перевод и склеивание битов этих чисел с последующим занесением результата в переменную res2, которая находится в сегменте ES. А затем этот результат заносим в сегмент DS. На рисунке 2 видно, что в сегменте DS по смещению 0000 (переменная res1) лежит число 83h. А это значит, что программа отработал верно. Результат программы и состояние регистров CPU можно увидеть на рисунках 2 и 3 соответственно.

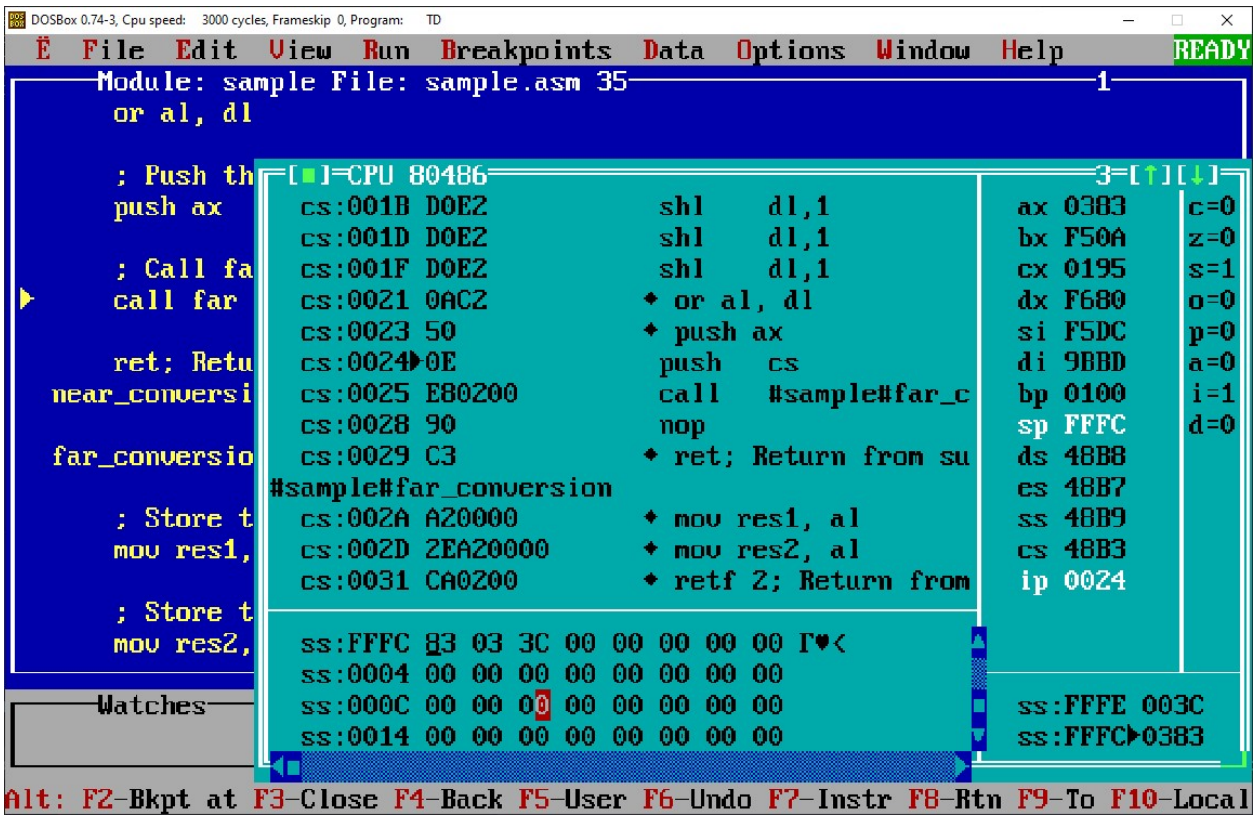


Рисунок 1 – Состояние сегмента стека по адресу SP после занесения значения переменной input при помощи команды push

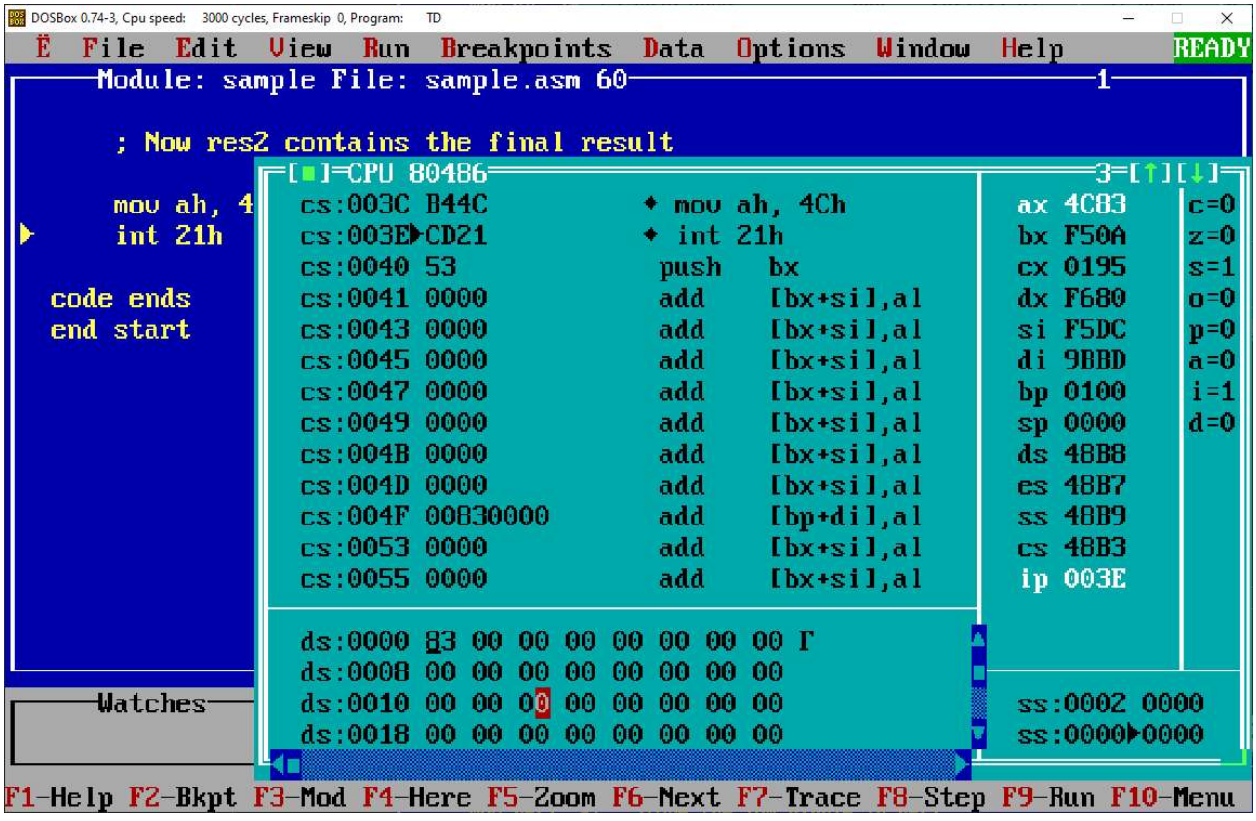


Рисунок 2 – Состояние сегмента DS (result) на момент завершения программы

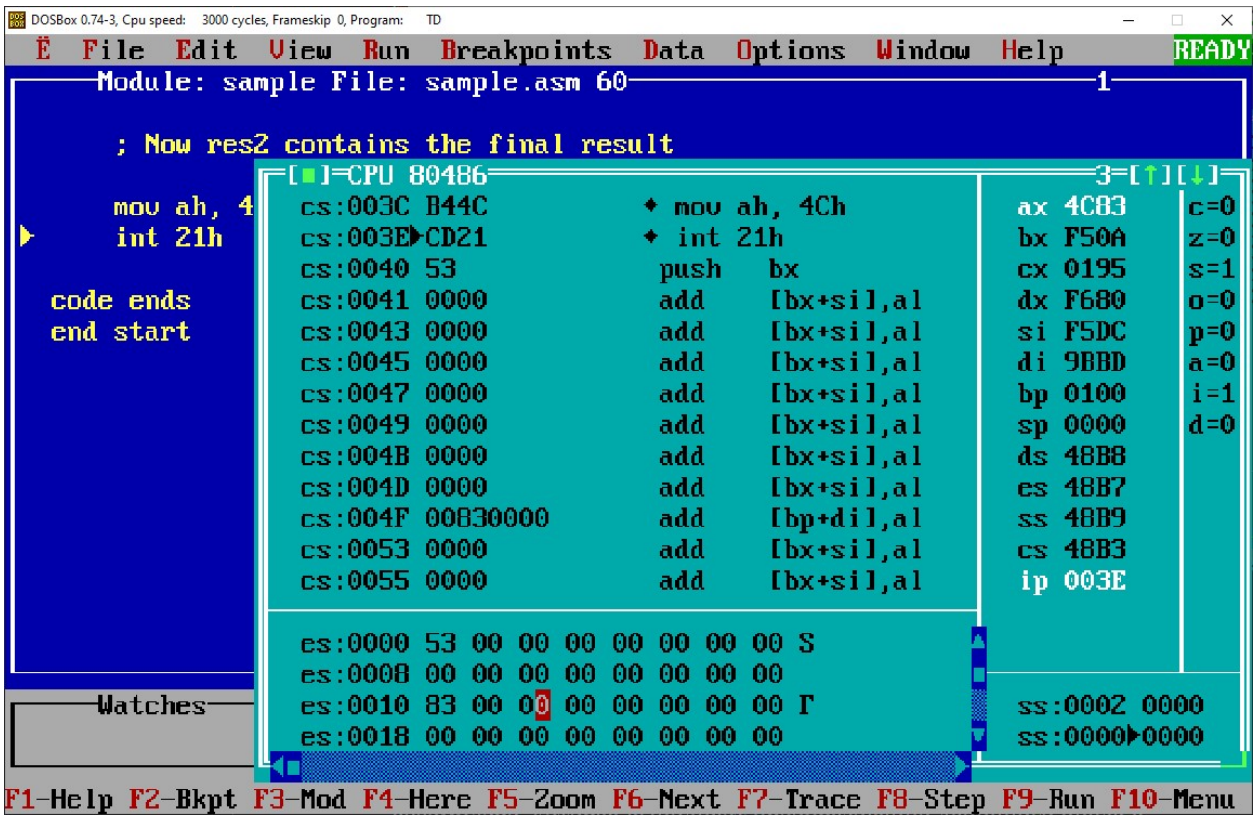


Рисунок 3 – Состояние сегмента ES (data) на момент завершения программы

## 5. Вывод

В ходе выполненной работы ознакомился с ближними и дальними подпрограммами, освоил работу со стеком.