**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 CISC-процессора.

**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 программе должен быть стек. Нужно, чтобы хотя бы один параметр некоторой передавался подпрограмме 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, 56, 73

data\_in ends

data\_out segment

res1 db 0

data\_out ends

stack segment

dw 100 dup(0) ; Stack definition

stack ends

code segment

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 ax

; Call far\_conversion from the far\_code segment

call far ptr far\_conversion

ret; Return from subroutine

near\_conversion endp

start:

mov ax, stack

mov ss, ax

call near\_conversion

mov ah, 4Ch

int 21h

code ends

far\_code segment

res2 db 0

assume CS:far\_code

far\_conversion proc far

; Store the low nibble in res1

mov res1, al

; Store the high nibble in res2

mov res2, al

retf 2; Far return '2' to pop ax

far\_conversion endp

far\_code ends

end start

**2. Листинг программы**

1 0029 B8B948 mov ax, stack 002C ax 48B9

2 002C 8ED0 mov ss, ax 0000 ss 48B9

3 0028 E8D5FF call near\_conversion

0000 sp FFFE

4 0000 B8B748 mov ax, data\_in

0003 ax 48B7

4 0003 8EC0 mov es, ax 0005 es 48B7

5 0005 B8B848 mov ax, data\_out

0008 ax 48B8

6 0008 8ED8 mov ds, ax 000A ds 48B8

7 000A 26A00000 mov al, input 000E ax 4853 = 53

8 000E 32E4 xor ah, ah 0010 ax 0053

9 0010 B30A mov bl, 10 0012 bx 0008

10 0012 F6F3 div bl 0014 ax 0308

11 0014 8AD0 mov dl, al 0016 dx 0008

12 0016 8AC4 mov al, ah 0018 ax 0303

13 0018 D0E2 shl dl, 1 001A dx 0010

14 001A D0E2 shl dl, 1 001C dx 0020

15 001C D0E2 shl dl, 1 001E dx 0040

16 001E D0E2 shl dl, 1 0020 dx 0080

17 0020 0AC2 or al, dl 0022 ax 0383

18 0022 50 push ax 0023 sp FFFC

19 0023 9A0100C648 call far ptr far\_conversion

0001 sp FFF8, cs 48C6

20 0001 A20000 mov res1, al 0004 ds:0000 = 83

21 0004 2EA20000 mov res2, al 0008 ds:0000 = 83

22 0008 CA0200 retf 2 0028 sp FFFE, cs 48B3

23 0028 C3 ret 0031 sp 0000

24 0031 B44C mov ah, 4Ch 0033 ax 4C83

25 0033 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 | 0029 | B8B948 | B8 | mov ax, stack | 002C | ax 48B9 |
| 2 | 002C | 8ED0 | 8ED0 | mov ss, ax | 0000 | ss 48B9 |
| 3 | 0028 | E8D5FF | E8D5FF | call near\_conversion | 0000 | sp FFFE |
| 4 | 0000 | B8B748 | B8 | mov ax, data\_in | 0003 | ax 48B7 |
| 4 | 0003 | 8EC0 | 8EC0 | mov es, ax | 0005 | es 48B7 |
| 5 | 0005 | B8B848 | B8 | mov ax, data\_out | 0008 | ax 48B8 |
| 6 | 0008 | 8ED8 | 8ED8 | mov ds, ax | 000A | ds 48B8 |
| 7 | 000A | 26A00000 | 26A0 | mov al, input | 000E | ax 4853 = 53 |
| 8 | 000E | 32E4 | 32E4 | xor ah, ah | 0010 | ax 0053 |
| 9 | 0010 | B30A | B30A | mov bl, 10 | 0012 | bx 0008 |
| 10 | 0012 | F6F3 | F6F3 | div bl | 0014 | ax 0308 |
| 11 | 0014 | 8AD0 | 8AD0 | mov dl, al | 0016 | dx 0008 |
| 12 | 0016 | 8AC4 | 8AC4 | mov al, ah | 0018 | ax 0303 |
| 13 | 0018 | D0E2 | D0E2 | shl dl, 1 | 001A | dx 0010 |
| 14 | 001A | D0E2 | D0E2 | shl dl, 1 | 001C | dx 0020 |
| 15 | 001C | D0E2 | D0E2 | shl dl, 1 | 001E | dx 0040 |
| 16 | 001E | D0E2 | D0E2 | shl dl, 1 | 0020 | dx 0080 |
| 17 | 0020 | 0AC2 | 0AC2 | or al, dl | 0022 | ax 0383 |
| 18 | 0022 | 50 | 50 | push ax | 0023 | sp FFFC |
| 19 | 0023 | 9A0100C648 | 9A | call far ptr\_far\_conversion | 0001 | sp FFF8, cs 48C6 |
| 20 | 0001 | A20000 | A2 | mov res1, al | 0004 | ds:0000 = 83 |
| 21 | 0004 | 2EA20000 | 2EA2 | mov res2, al | 0008 | ds:0000 = 83 |
| 22 | 0008 | CA0200 | CA | retf 2 | 0028 | sp FFFE, cs 48B3 |
| 23 | 0028 | C3 | C3 | ret | 0031 | sp 0000 |
| 24 | 0031 | B44C | B4 | mov ah, 4Ch | 0033 | ax 4C83 |
| 25 | 0033 | CD21 | CD | 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 массив чисел 8, 3. Далее в ближней подпрограмме эти числа заносятся в сегмент ES. После этого происходит перевод и склеивание битов этих чисел с последующим занесением результата в переменную res2, которая находится в сегменте ES. А затем этот результат в дальней подпрограмме 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 1, 2 и 3 соответственно.

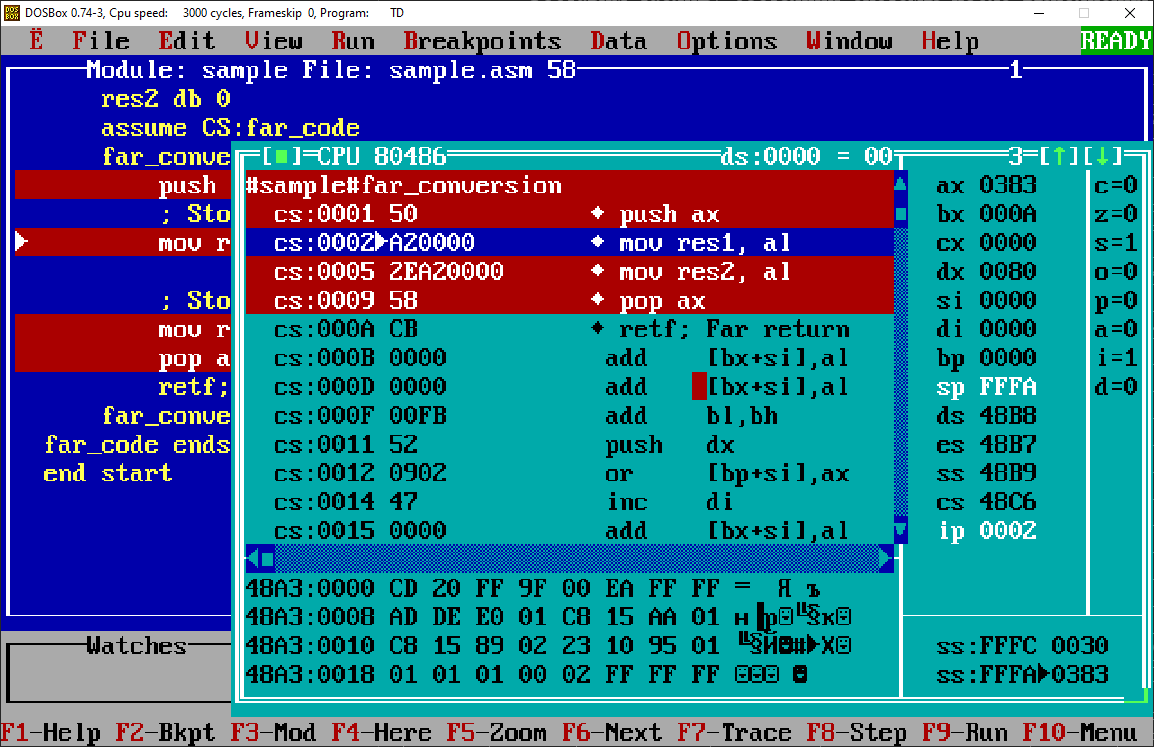


Рисунок 1 – Состояние сегмента SS по адресу SP после занесения значения регистра ax в стек при помощи команды push

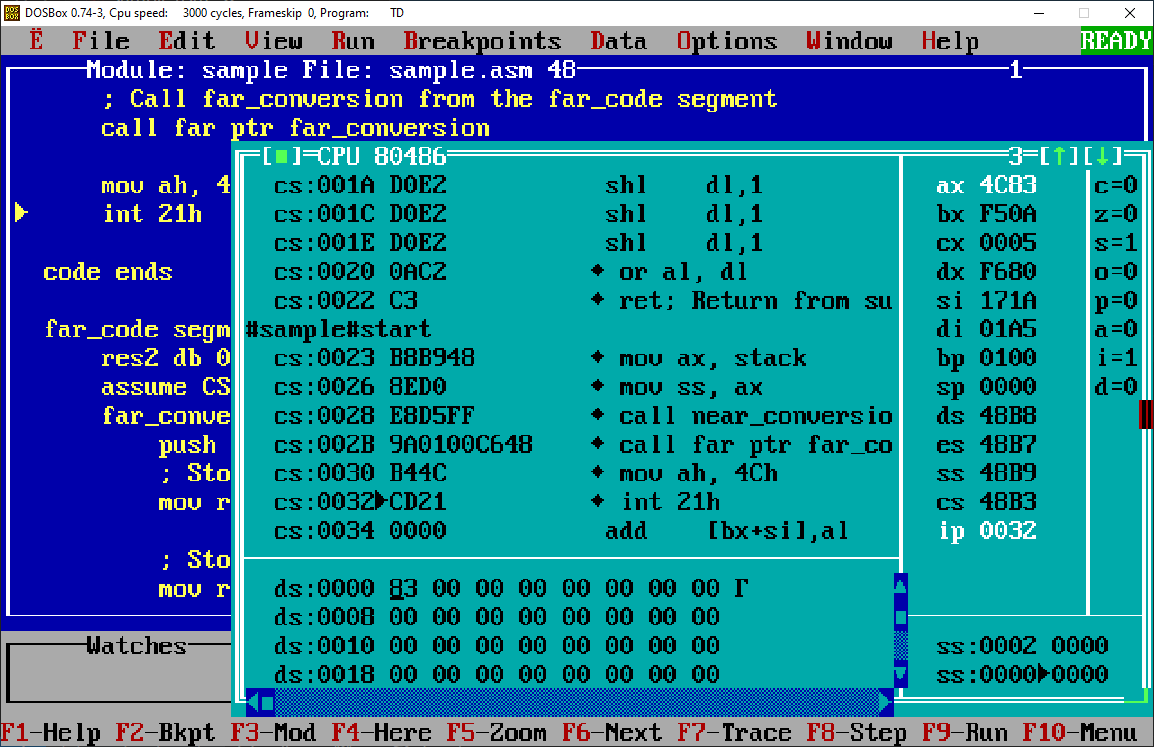


Рисунок 2 – Состояние сегмента DS (result) на момент завершения программы

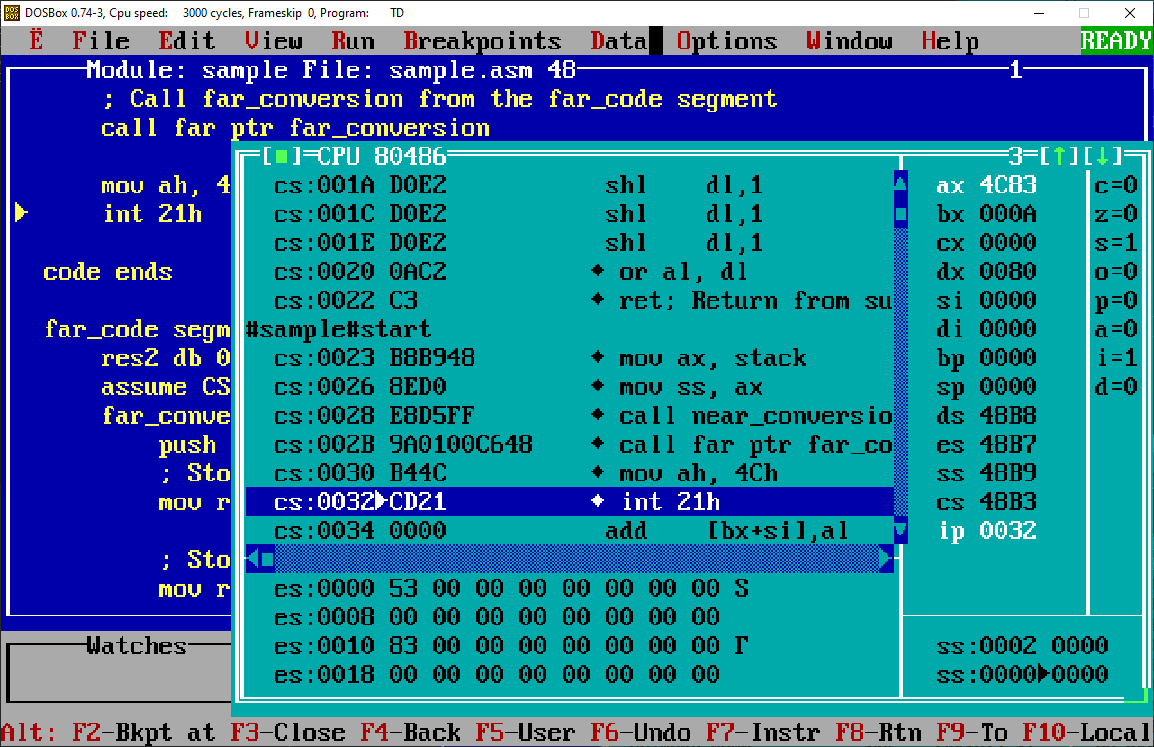


Рисунок 3 – Состояние сегмента ES (data) на момент завершения программы

**5. Вывод**

В ходе выполненной работы ознакомился с ближними и дальними подпрограммами, освоил работу со стеком. Изучил основы устройства и принципов работы компьютера принстонской архитектуры CISC-процессора.