**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, 99

data\_in ends

data\_out segment

res1 db 0

res2 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 si, 0 ; Initialize index for array traversal

convert\_loop:

mov al, input[si] ; Load the current element from the array

xor ah, ah

mov bl, 10

div bl

mov dl, al

mov al, ah

shl dl, 4

or al, dl

; Check the index to determine which element is being processed

cmp si, 0

je store\_res1 ; Jump to store\_res1 if si is 0

mov res2, al

push ax ; push 99 to stack

call far ptr far\_conversion

jmp next\_iteration

store\_res1:

mov res1, al

next\_iteration:

; Move to the next element in the array

inc si

cmp si, 2 ; Check if all elements are processed

jl convert\_loop

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

res3 db 0

assume CS:far\_code

far\_conversion proc far

mov res3, al

retf 2

far\_conversion endp

far\_code ends

end start

**2. Листинг программы**

1. 0041: B8 B748 ; mov ax, data\_out

2. 0044: 8ED8 ; mov ds, ax

3. 0046: BE 0000 ; mov si, 0

4. 0049: 26 8A840000 ; mov al, input[si]

5. 004E: 32 E4 ; xor ah, ah

6. 0050: B3 0A ; mov bl, 10

7. 0052: F6 F3 ; div bl

8. 0054: 8A D0 ; mov dl, al

9. 0056: 8A C4 ; mov al, ah

10. 0058: D0 E2 ; shl dl, 1

11. 005A: D0 E2 ; shl dl, 1

12. 005C: D0 E2 ; shl dl, 1

13. 005E: D0 E2 ; shl dl, 1

14. 0060: 0A C2 ; or al, dl

15. 0062: 83 FE 00 ; cmp si, 0

16. 0065: 74 0C ; je store\_res1

17. 0067: A2 0000 ; mov res1, al

18. 006A: 46 ; inc si

19. 006B: 83 FE 02 ; cmp si, 2

20. 006E: 7C CD ; jl convert\_loop

21. 0070: 26 8A840000 ; mov al, input[si]

22. 0075: 32 E4 ; xor ah, ah

23. 0077: B3 0A ; mov bl, 10

24. 0079: F6 F3 ; div bl

25. 007B: 8A D0 ; mov dl, al

26. 007D: 8A C4 ; mov al, ah

27. 007F: D0 E2 ; shl dl, 1

28. 0081: D0 E2 ; shl dl, 1

29. 0083: D0 E2 ; shl dl, 1

30. 0085: D0 E2 ; shl dl, 1

31. 0087: 0A C2 ; or al, dl

32. 0089: 83 FE 00 ; cmp si, 0

33. 008C: 74 0C ; je store\_res1

34. 008E: A2 0100 ; mov res2, al

35. 0091: 50 ; push ax

36. 0092: 9A 0100C748 ; call far ptr far\_conversion

37. 0097: 2E A20000 ; mov res3, al

38. 009B: CA 0200 ; retf 2

39. 009E: EB 04 ; jmp next\_iteration

40. 00A0: 46 ; inc si

41. 00A1: 83 FE 02 ; cmp si, 2

42. 00A4: 7C CD ; jl convert\_loop

43. 00A6: C3 ; ret

44. 00A7: B4 4C ; mov ah, 4Ch

45. 00A9: CD 21 ; 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 | 0041 | B8BA48 | B8 | mov ax, stack | 0044 | ax 48BA |
| 2 | 0044 | 8ED0 | 8ED0 | mov ss, ax | 0046 | ss 48BA |
| 3 | 0046 | E8B7FF | E8 | call near\_conversion | 0000 | sp FFFE ss:sp -> 0049 |
| 4 | 0000 | B8B848 | B8 | mov ax, data\_in | 0003 | ax 48B8 |
| 4 | 0003 | 8EC0 | 8EC0 | mov es, ax | 0005 | es 48B8 |
| 5 | 0005 | B8B848 | B8 | mov ax, data\_out | 0008 | ax 48B9 |
| 6 | 0008 | 8ED8 | 8ED8 | mov ds, ax | 000A | ds 48B9 |
| 7 | 000A | BE0000 | BE | mov si, 0 | 000D | si 0000 |
| 8 | 000D | 268A840000 | 268A | mov al, input[si] | 0012 | ax 4853 |
| 9 | 0012 | 32E4 | 32E4 | xor ah, ah | 0014 | ax 0053 |
| 10 | 0014 | B30A | B30A | mov bl, 10 | 0016 | bx F60A |
| 11 | 0016 | F6F3 | F6F3 | div bl | 0018 | ax 0308 |
| 12 | 0018 | 8AD0 | 8AD0 | mov dl, al | 001A | dx B408 |
| 13 | 001A | 8AC4 | 8AC4 | mov al, ah | 001C | ax 0303 |
| 14 | 001C | D0E2 | D0E2 | shl dl, 1 | 001E | dx 0010 |
| 15 | 001E | D0E2 | D0E2 | sh dl, 1 | 0020 | dx 0020 |
| 16 | 0020 | D0E2 | D0E2 | shl dl, 1 | 0022 | dx 0040 |
| 17 | 0022 | D0E2 | D0E2 | shl dl, 1 | 0024 | dx 0080 |
| 18 | 0024 | 0AC2 | 0AC2 | or al, dl | 0026 | ax 0383 |
| 19 | 0026 | 83FE00 | 83FE00 | cmp si, 0 | 0029 |  |
| 20 | 0029 | 740C | 740C | je store\_res1 | 0037 |  |
| 21 | 0037 | A20000 | A2 | mov res1, al | 003A | ds:0000 = 83 |
| 22 | 003A | 46 | 46 | inc si | 003B | si 0001 |
| 23 | 003B | 83FE02 | 83FE | cmp si, 2 | 003E |  |
| 24 | 003E | 7CCD | 7CCD | jl convert\_loop | 000D |  |
| 25 | 000D | 268A840000 | 268A | mov al, input[si] | 0012 | ax 0363 |
| 26 | 0012 | 32E4 | 32E4 | xor ah, ah | 0014 | ax 0063 |
| 27 | 0014 | B30A | B30A | mov bl, 10 | 0016 |  |
| 28 | 0016 | F6F3 | F6F3 | div bl | 0018 | ax 0909 |
| 29 | 0018 | 8AD0 | 8AD0 | mov dl, al | 001A | dx 0009 |
| 30 | 001A | 8AC4 | 8AC4 | mov al, ah | 001C |  |
| 31 | 001C | D0E2 | D0E2 | shl dl, 1 | 001E | dx 0012 |
| 32 | 001E | D0E2 | D0E2 | shl dl, 1 | 0020 | dx 0024 |
| 33 | 0020 | D0E2 | D0E2 | shl dl, 1 | 0022 | dx 0048 |
| 34 | 0022 | D0E2 | D0E2 | shl dl, 1 | 0024 | dx 0090 |
| 35 | 0026 | 0AC2 | 0AC2 | or al, dl | 0026 | ax 0999 |
| 36 | 0026 | 83FE00 | 83FE | cmp si, 0 | 0029 |  |
| 37 | 0029 | 740C | 740C | je store\_res1 | 002B |  |
| 38 | 002B | A20100 | A2 | mov res2, al | 002E |  |
| 39 | 002E | 50 | 50 | push ax | 002F | sp FFFC, ss:sp -> 0999 |
| 40 | 002F | 9A0100C748 | 9A01 | call far ptr far\_co | 0001 | sp FFF8, cs 48C7, ss:sp->0034 |
| 41 | 0001 | 2EA20000 | 2EA2 | mov res3, al | 0005 |  |
| 42 | 0005 | CA0200 | CA02 | retf 2 | 0034 | sp FFFE, cs 48B3, ss:sp -> 0049 |
| 43 | 0034 | EB04 | EB04 | jmp | 003A |  |
| 44 | 003A | 46 | 46 | inc si | 003B | si 0002 |
| 45 | 003B | 83FE02 | 83FE02 | cmp si, 2 | 003E |  |
| 46 | 003E | 7CCD | 7CCD | jl convert\_loop | 0040 |  |
| 47 | 0040 | C3 | C3 | ret | 0049 | sp 0000, ss:sp -> 0000 |
| 48 | 0049 | B44C | B4 | mov ah, 4Ch | 004B | ax 4C99 |
| 49 | 004B | 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 83, 99. Далее в ближней подпрограмме первое число заносится в сегмент ES. После этого происходит перевод и склеивание битов этих чисел с последующим занесением результата в переменную res1, которая находится в сегменте ES. А затем этот результат в дальней подпрограмме заносим в сегмент DS. Для второго числа выполняется та же самая циклическая операция. На рисунке 2 видно, что в сегменте DS по смещению 0000 (переменная res1) лежит число 83h, 99h. А это значит, что программа отработал верно. Результат программы и состояние регистров CPU можно увидеть на рисунках 1, 2 и 3 соответственно.

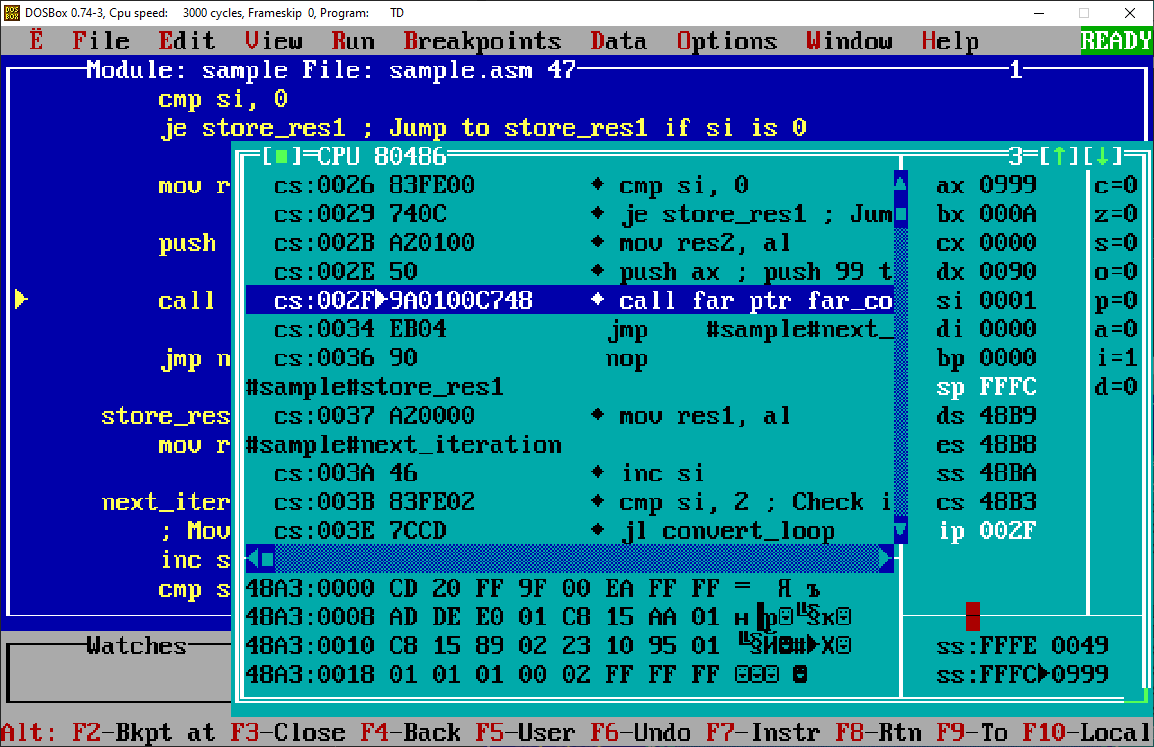


Рисунок 1 – Состояние сегмента SS по адресу SP после занесения значения регистра ax в стек при помощи команды push

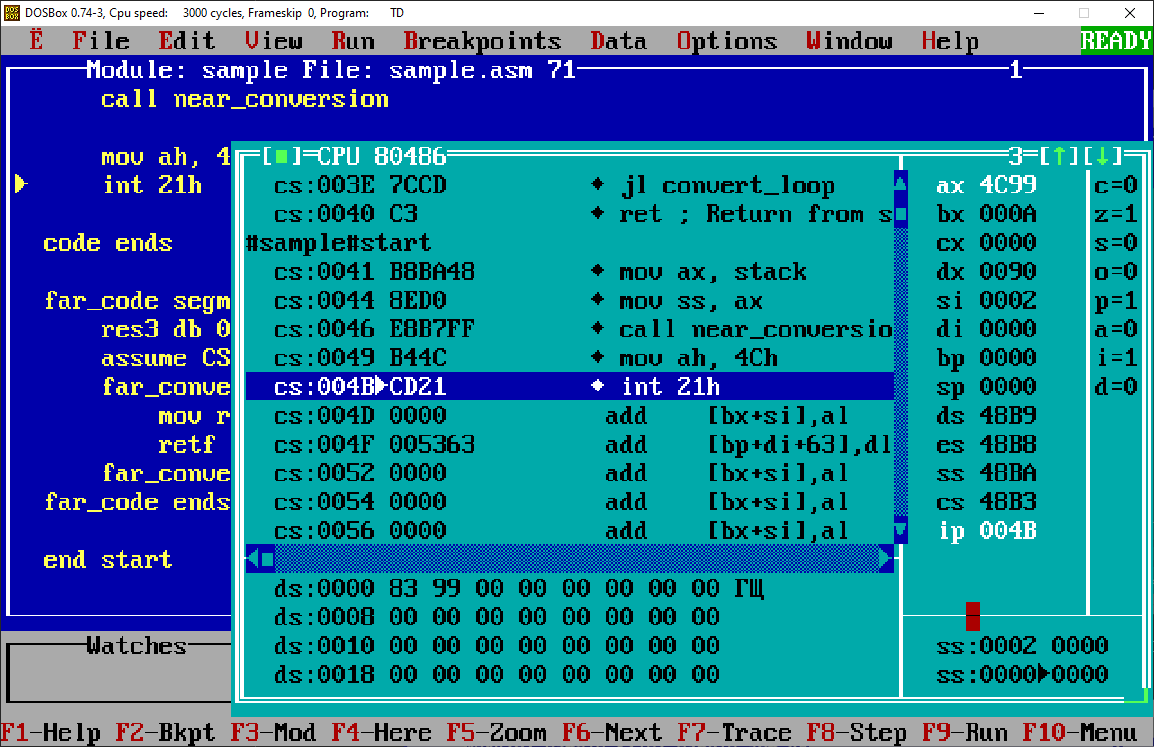


Рисунок 2 – Состояние сегмента DS (result) на момент завершения программы

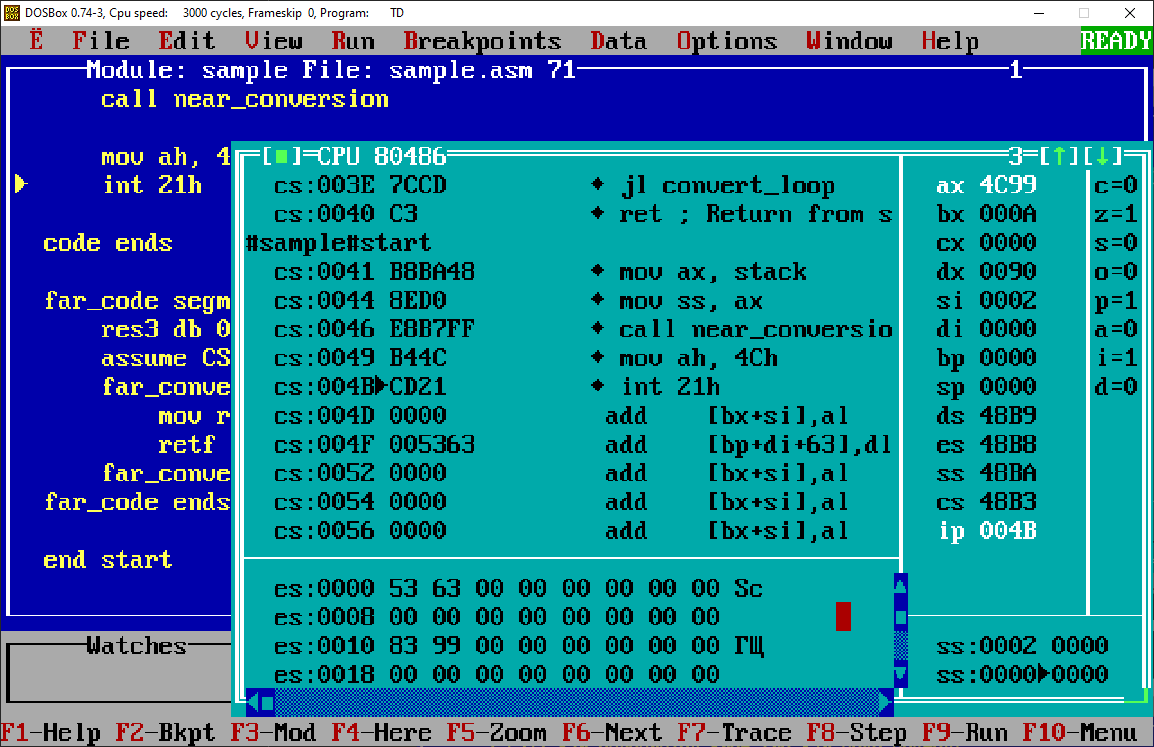


Рисунок 3 – Состояние сегмента ES (data) на момент завершения программы

**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.