МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Технічна Кібернетика»

**Лабораторна робота №2**

**з дисципліни «архітектура комп’ютера»**

**на тему: «Команди пересилання та обміну даними в мові Асемблера»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.І.

Дніпро, 2019

**Тема.** Команди пересилання та обміну даними в мові Ассемблера.

**Мета.** Вивчити принципи організації пам’яті комп’ютера, адресації даних. Ознайомитись з поняттям ефективної, логічної і фізичної адреси. Вивчити регістри центрального процесора, їх призначення і взаємозв’язок. Вивчити команди пересилання і обміну даними, набути практичних навичок їх використання.

***1.Задача:***

Розробити програму для обробки інформації, що міститься в сегменті даних:

1) змінити текст згідно завданню обробки тексту (колонка «Обробка тексту» в таблиці);

2) записати в пам’ять логічну адресу (сегментну адресу і зміщення) другого числа (колонка «Змінні»);

3) переставити місцями старшу і молодшу частини чисел (колонка «Змінні») у форматі слова (двома способами) і подвійного слова (одним способом). Всі три способи обміну частин чисел повинні бути різними (команди пересилання, команда обміну, стек). Результати записати в нові змінні.

Таблиця 1

|  |  |  |
| --- | --- | --- |
| № | Змінні | Обробка тексту |
| 18 | text        db ‘Don’t bite off more than you can chew.’  numbers dd -193758                db 222                dw 45005 | Поміняти місцями слова «more» і «than». |

***2. Лістинг програми;***

Turbo Assembler Version 3.0 10/25/19 19:21:32 Page 1

2v5.asm

1 ﻿;NAME CODE OPERAND

2

3 ;сегмент стека

4 0000 sseg segment stack 'stack'

5 0000 80\*(??) db 128 dup(?)

6 0080 sseg ends

7

8 ; DATA WRITING TO SEGMENT

9 0000 dseg segment 'data'

10

11 ; текст

12

13 0000 44 6F 6E 27 74 20 62+ text db "Don't bite off more than you can chew.";38bytes(0-37)

14 69 74 65 20 6F 66 66+

15 20 6D 6F 72 65 20 74+

16 68 61 6E 20 79 6F 75+

17 20 63 61 6E 20 63 68+

18 65 77 2E

19

20 0026 FFFD0B22 numbers dd -193758;0 offset

21 002A DE db 222;4 bytes offset

22 002B AFCD dw 45005;5 bytes offset

23

24 002D ???????? ef\_ad dd ?;physical address components of 222 in numbers\*

25 0031 ???????? A dd ?;new number1

26 0035 ???? B dw ?;new number3

27 0037 ???? C dw ?;new number3v2

28

29 0039 dseg ends ;DATA WRITING END

30

31 ;CODE SEGMENT

32 0000 cseg segment 'code'

33 assume cs:cseg, ds:dseg, ss:sseg

34

35 ;мітка початку програми з ім’ям start

36 0000 B8 0000s start: mov ax, dseg

37 0003 8E D8 mov ds, ax

38

39 ;body

40 0005 FF 36 0011r push word ptr [text+17];"re" to stack

41 0009 FF 36 000Fr push word ptr [text+15];"mo" to stack

42 000D FF 36 0016r push word ptr [text+22];"an" to stack

43 0011 FF 36 0014r push word ptr [text+20];"th" to stack

44

45 0015 8F 06 000Fr pop word ptr [text+15];"th" to A+15

46 0019 8F 06 0011r pop word ptr [text+17];"an" to A+17

47 001D 8F 06 0014r pop word ptr [text+20];"mo" to A+20

48 0021 8F 06 0016r pop word ptr [text+22];"re" to A+22 ~end of task1

49

50 0025 B8 0028r lea ax, [numbers+2];/effective address of 222 in numbers/ to ax

51 0028 8C 1E 002Dr mov word ptr [ef\_ad], ds

52 002C A3 002Fr mov word ptr [ef\_ad+2], ax

53

54 002F FF 36 0028r FF 36 + push dword ptr [numbers]

55 0026r

56 0037 FF 36 002Ar FF 36 + push dword ptr [numbers+2]

57 0028r

Turbo Assembler Version 3.0 10/25/19 19:21:32 Page 2

2v5.asm

58 003F 8F 06 0031r 8F 06 + pop dword ptr [A]

59 0033r

60 0047 8F 06 0033r 8F 06 + pop dword ptr [A+2]; end of task 3.1

61 0035r

62

63 004F A0 002Br mov al, byte ptr [numbers+5]

64 0052 8A 26 002Cr mov ah, byte ptr [numbers+6]

65 0056 A2 0036r mov byte ptr [B+1], al

66 0059 88 26 0035r mov byte ptr [B], ah; end of task 3.2

67

68 005D A1 002Br mov ax, word ptr [numbers+5]

69 0060 86 C4 xchg al, ah

70 0062 A3 0037r mov C, ax; end of task 3.3

71

72 ;повернення керування ОС

73 0065 B4 4C mov ah, 4Ch

74 0067 CD 21 int 21h

75 0069 cseg ends

76 end start ;END

Turbo Assembler Version 3.0 10/25/19 19:21:32 Page 3

Symbol Table

Symbol Name Type Value

??DATE Text "10/25/19"

??FILENAME Text "2v5 "

??TIME Text "19:21:32"

??VERSION Number 0300

@CPU Text 0101H

@CURSEG Text CSEG

@FILENAME Text 2V5

@WORDSIZE Text 2

A Dword DSEG:0031

B Word DSEG:0035

C Word DSEG:0037

EF\_AD Dword DSEG:002D

NUMBERS Dword DSEG:0026

START Near CSEG:0000

TEXT Byte DSEG:0000

Groups & Segments Bit Size Align Combine Class

CSEG 16 0069 Para none CODE

DSEG 16 0039 Para none DATA

SSEG 16 0080 Para Stack STACK

***3. Аналіз результатів виконання програми на основі скрінів екрану;***

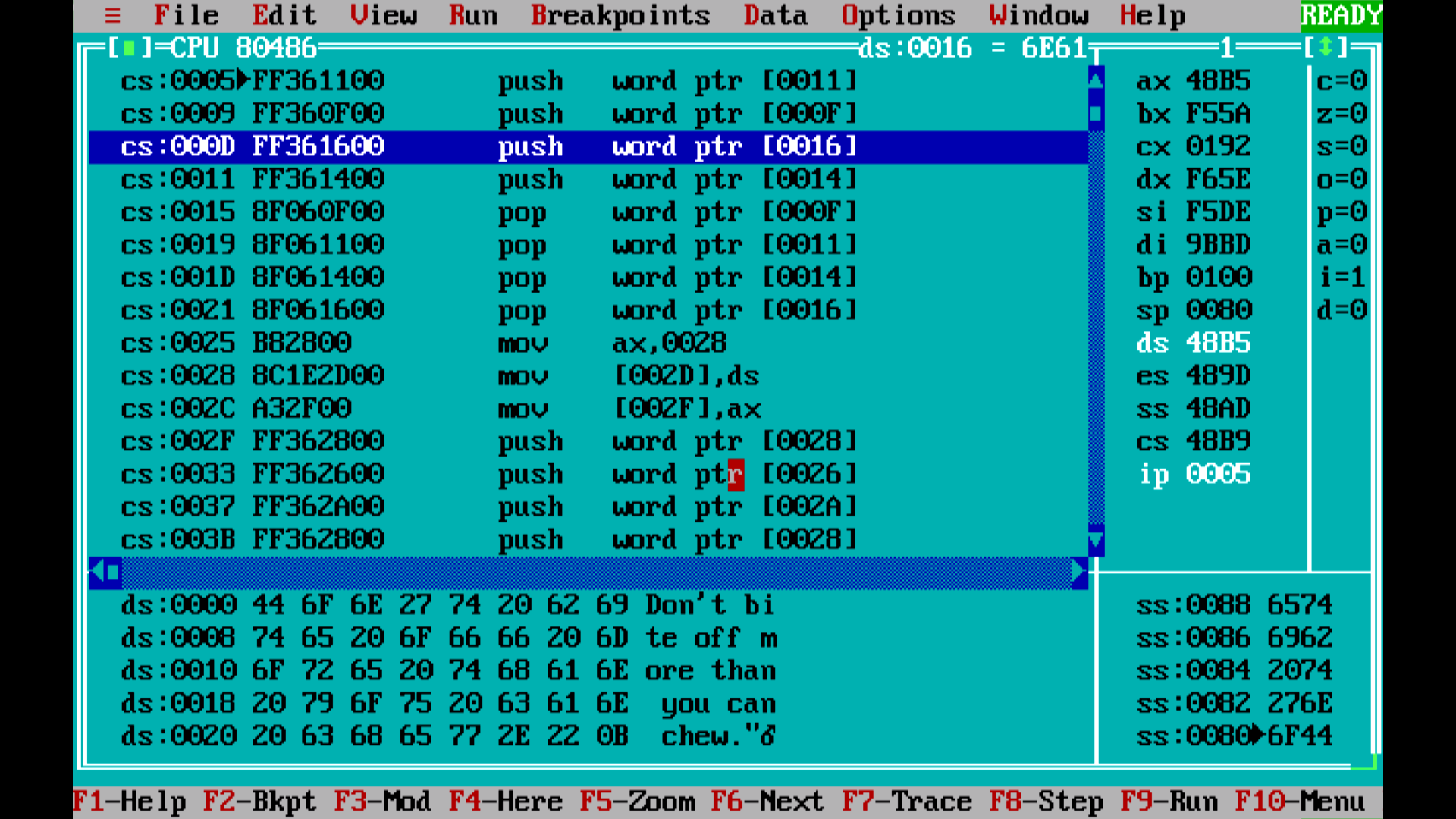
******

Рисунок 1

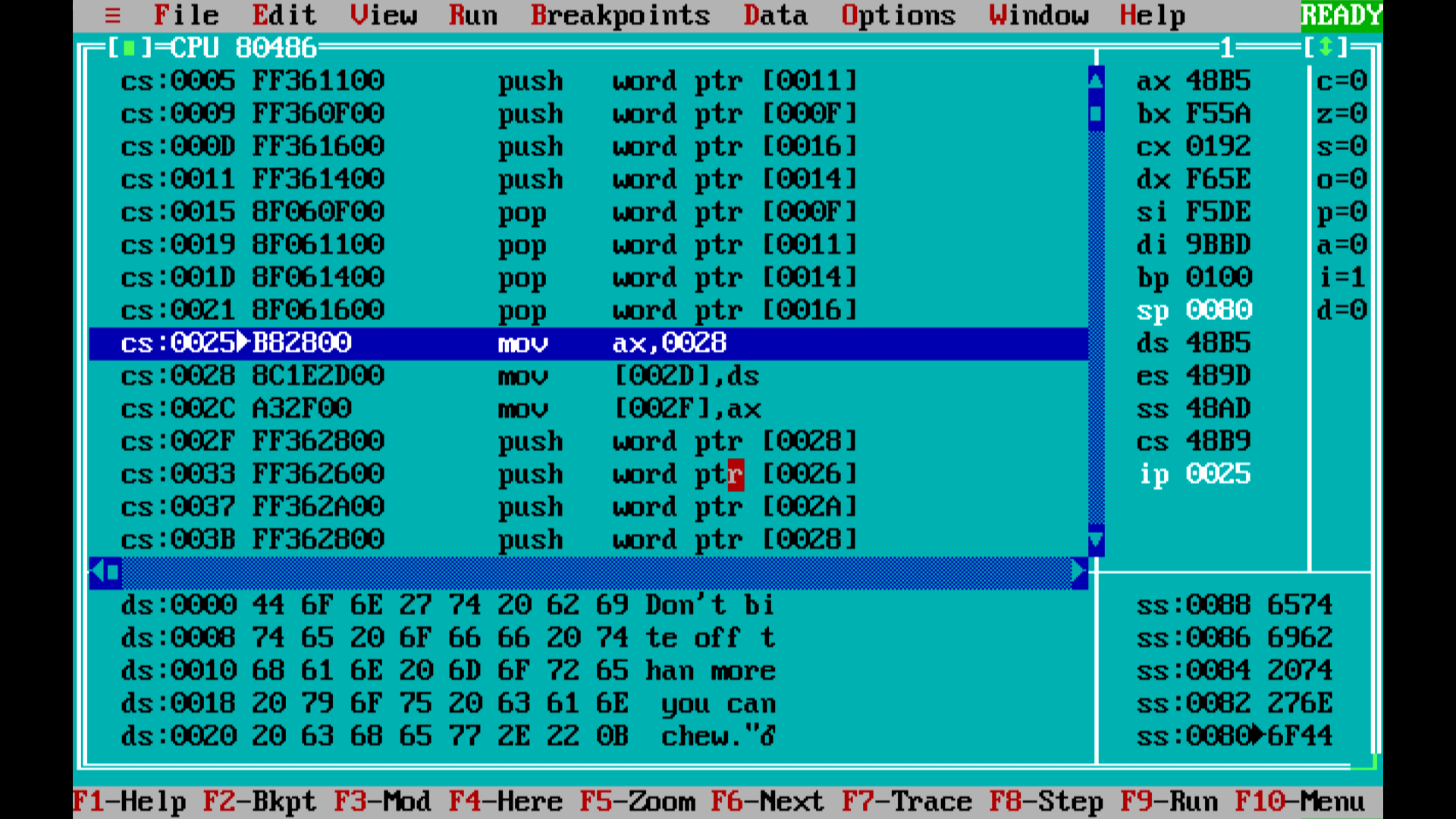
******

Рисунок 2

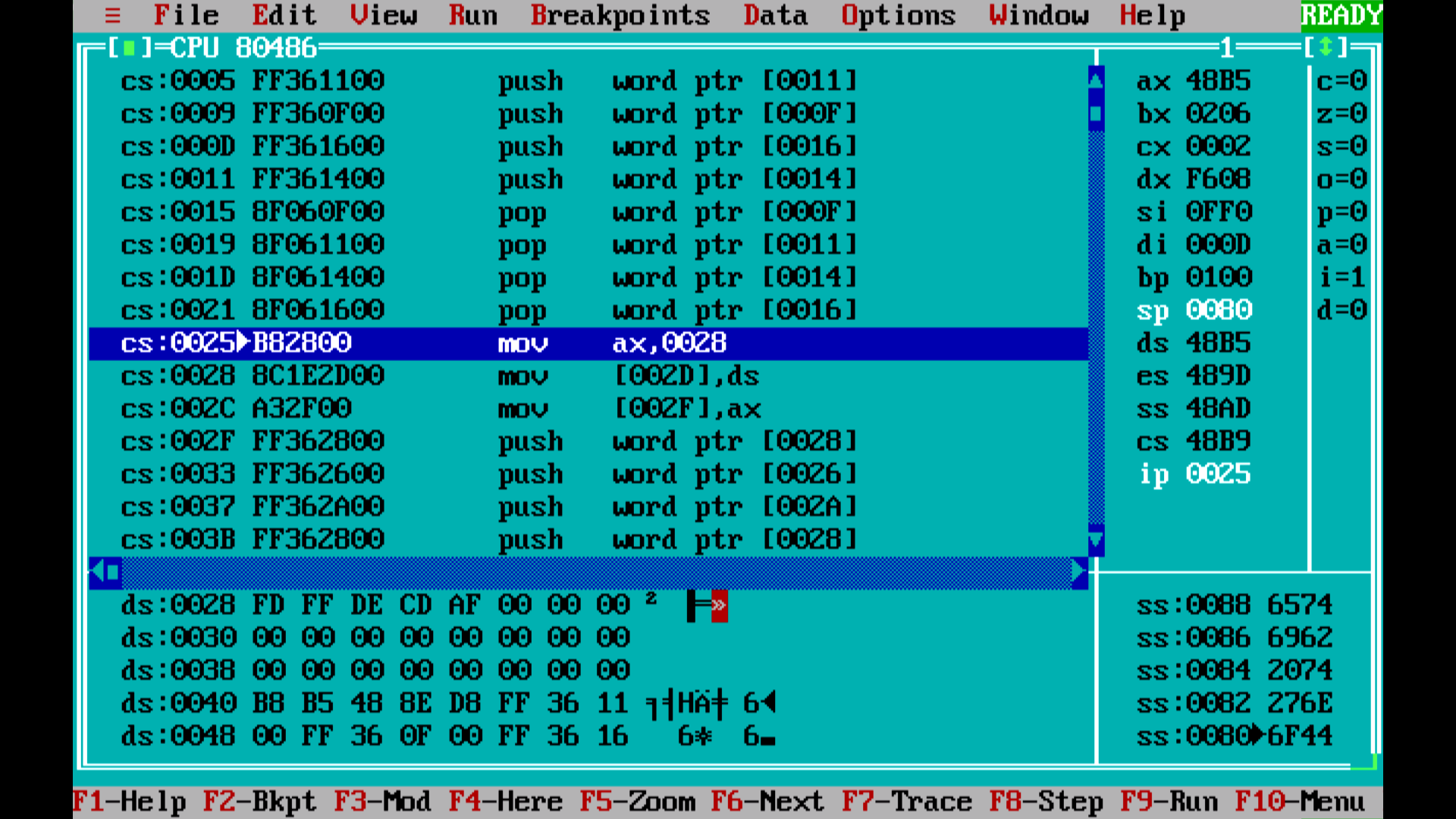
******

Рисунок 3

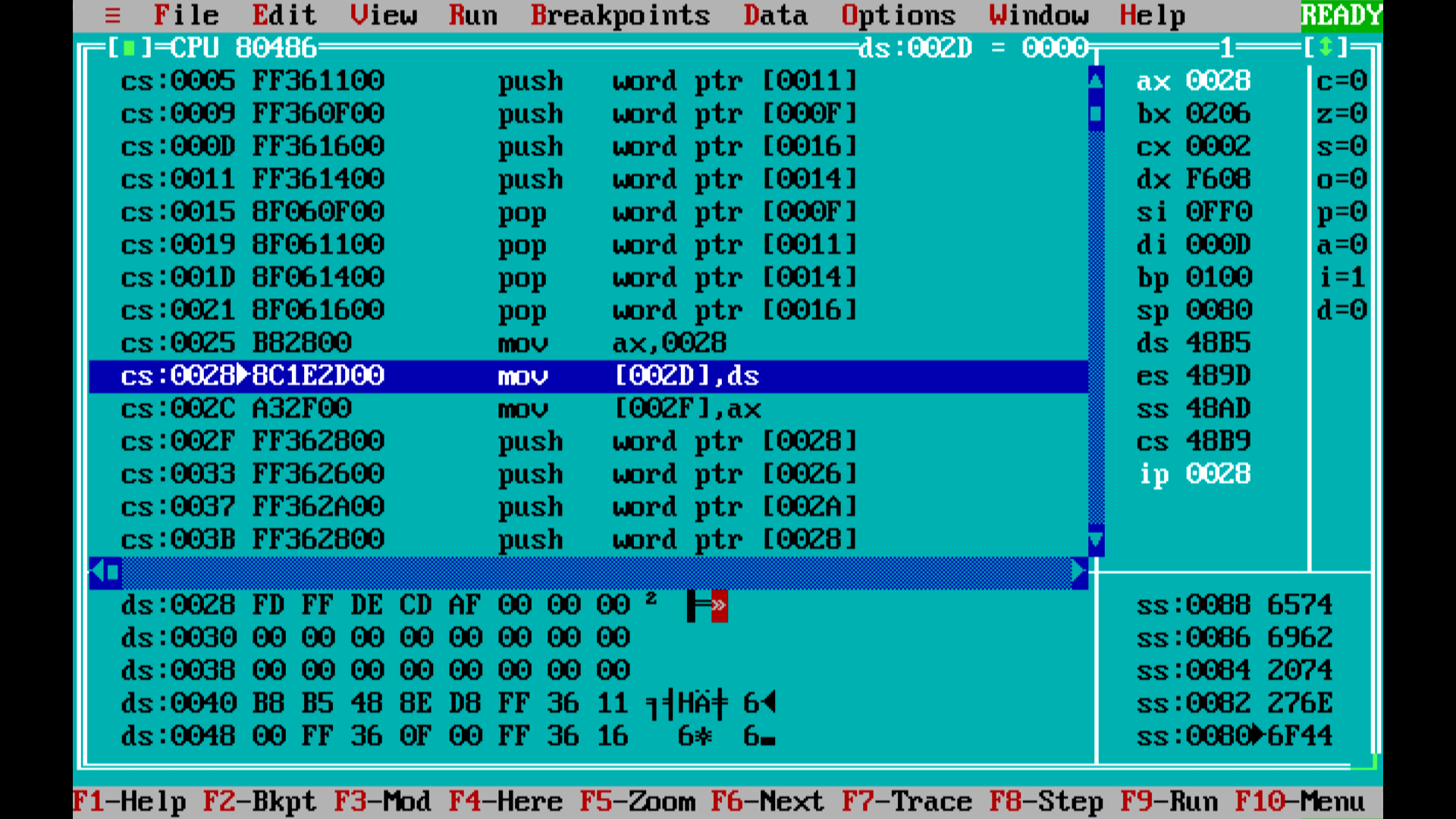
******

Рисунок 4

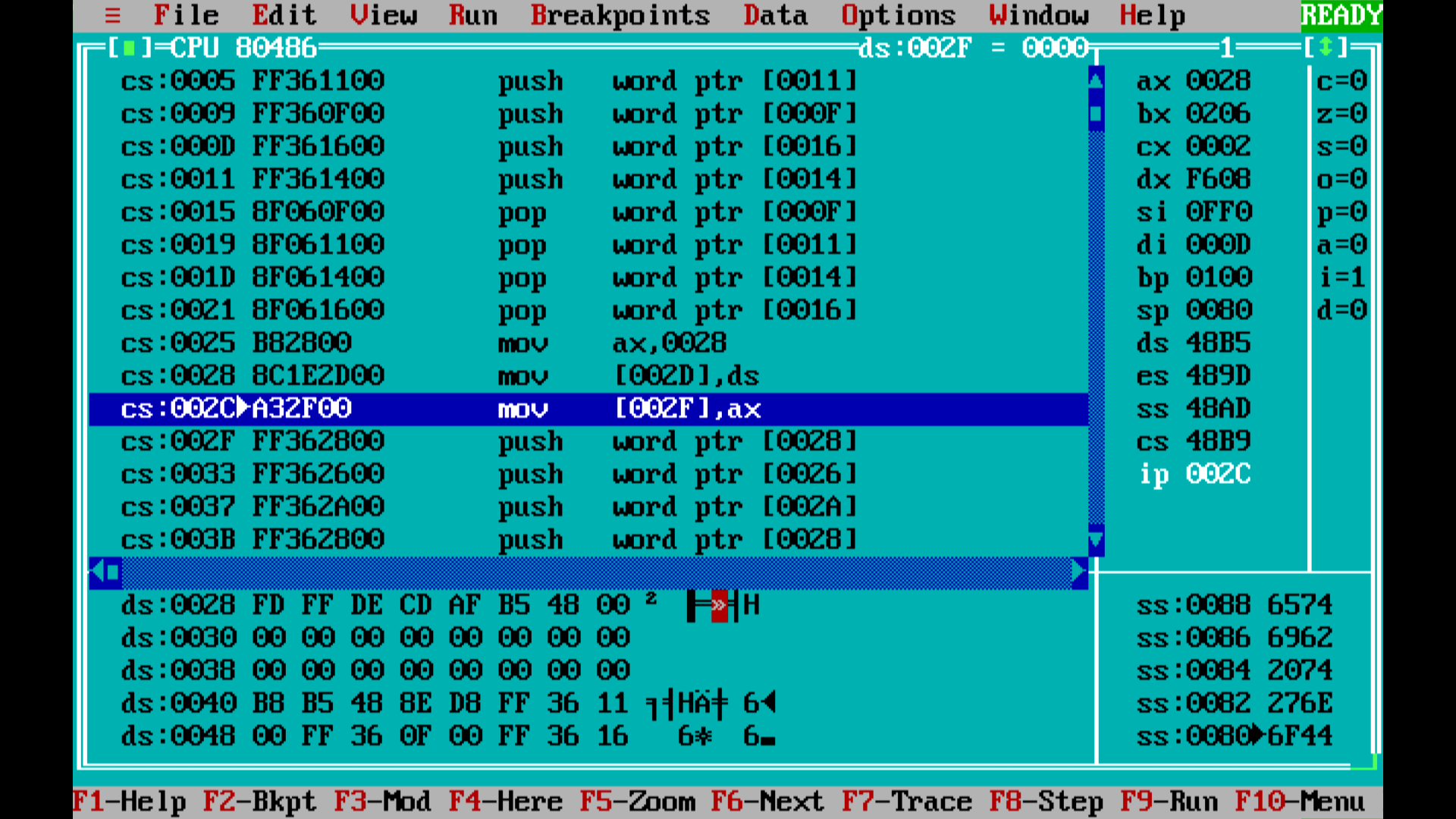
******

Рисунок 5

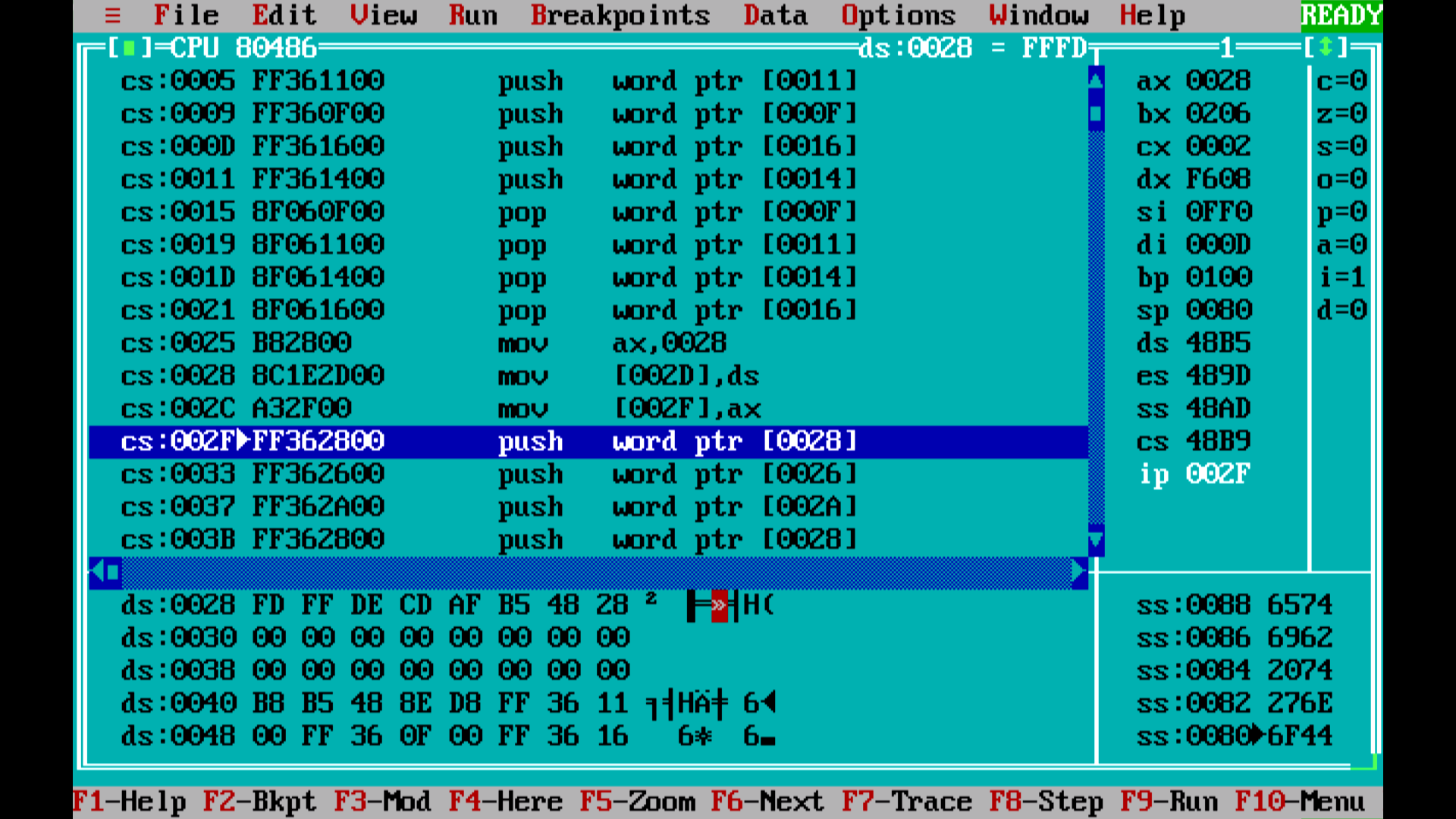
******

Рисунок 6

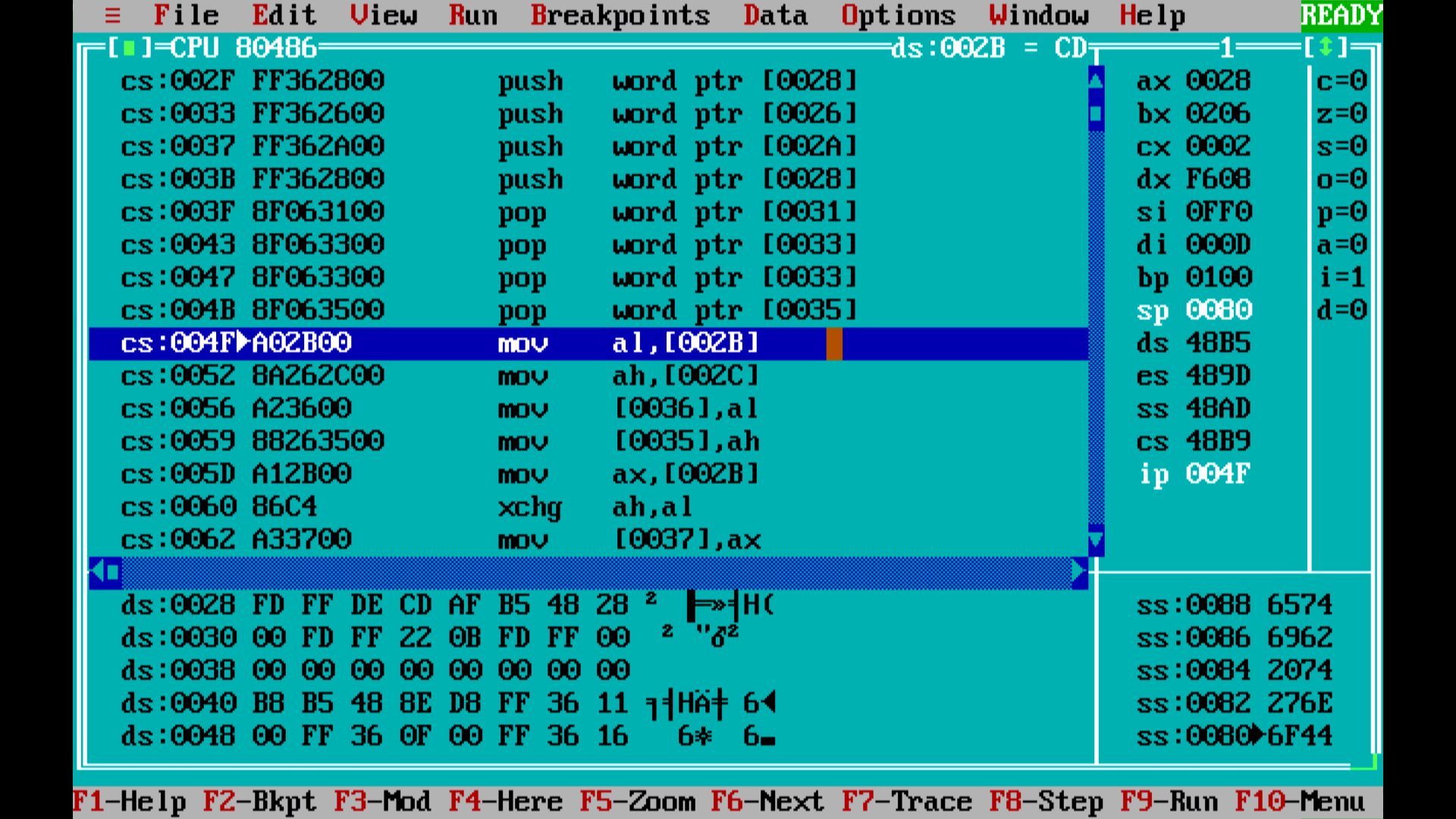
******

Рисунок 7

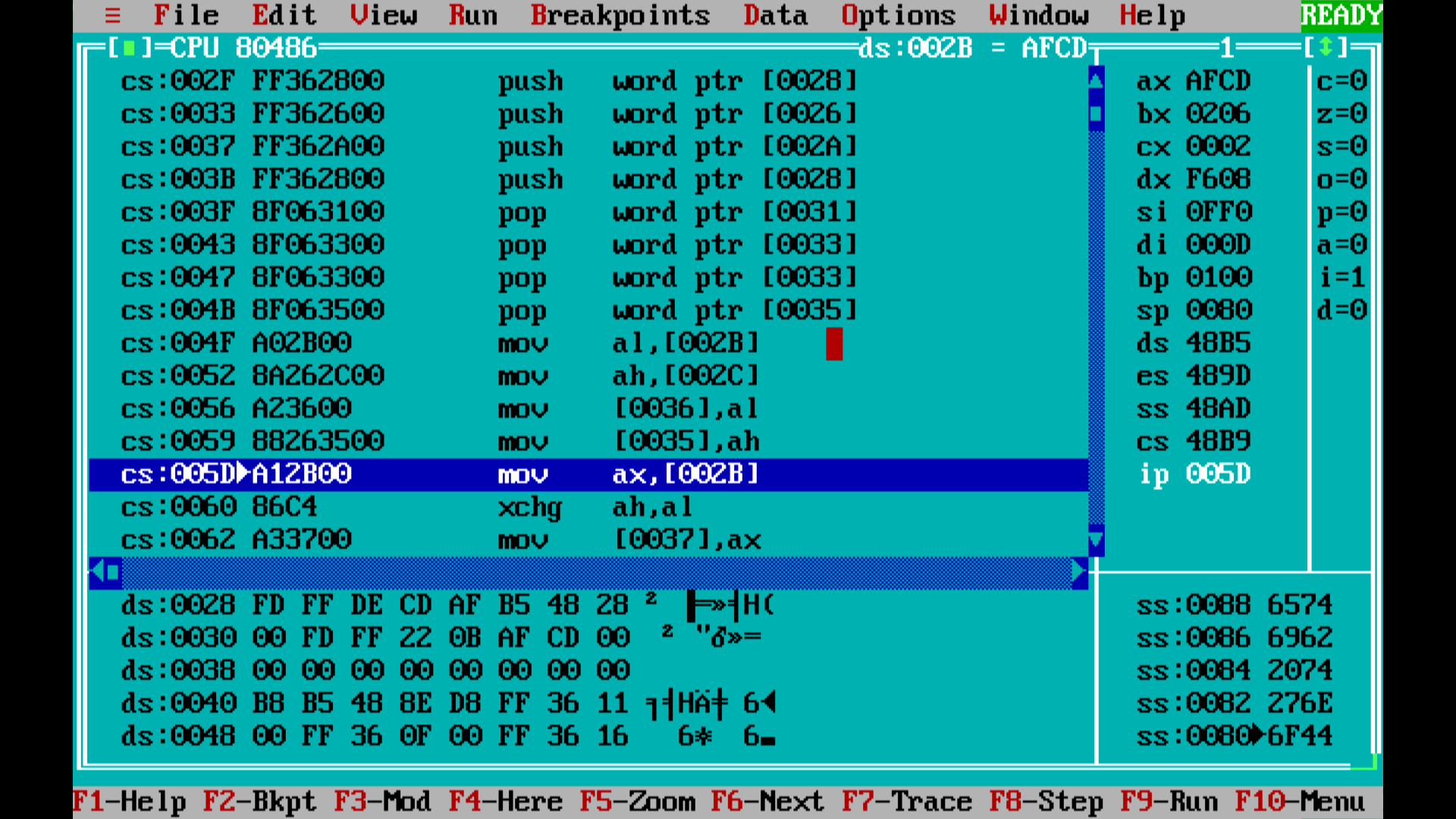
******

Рисунок 8

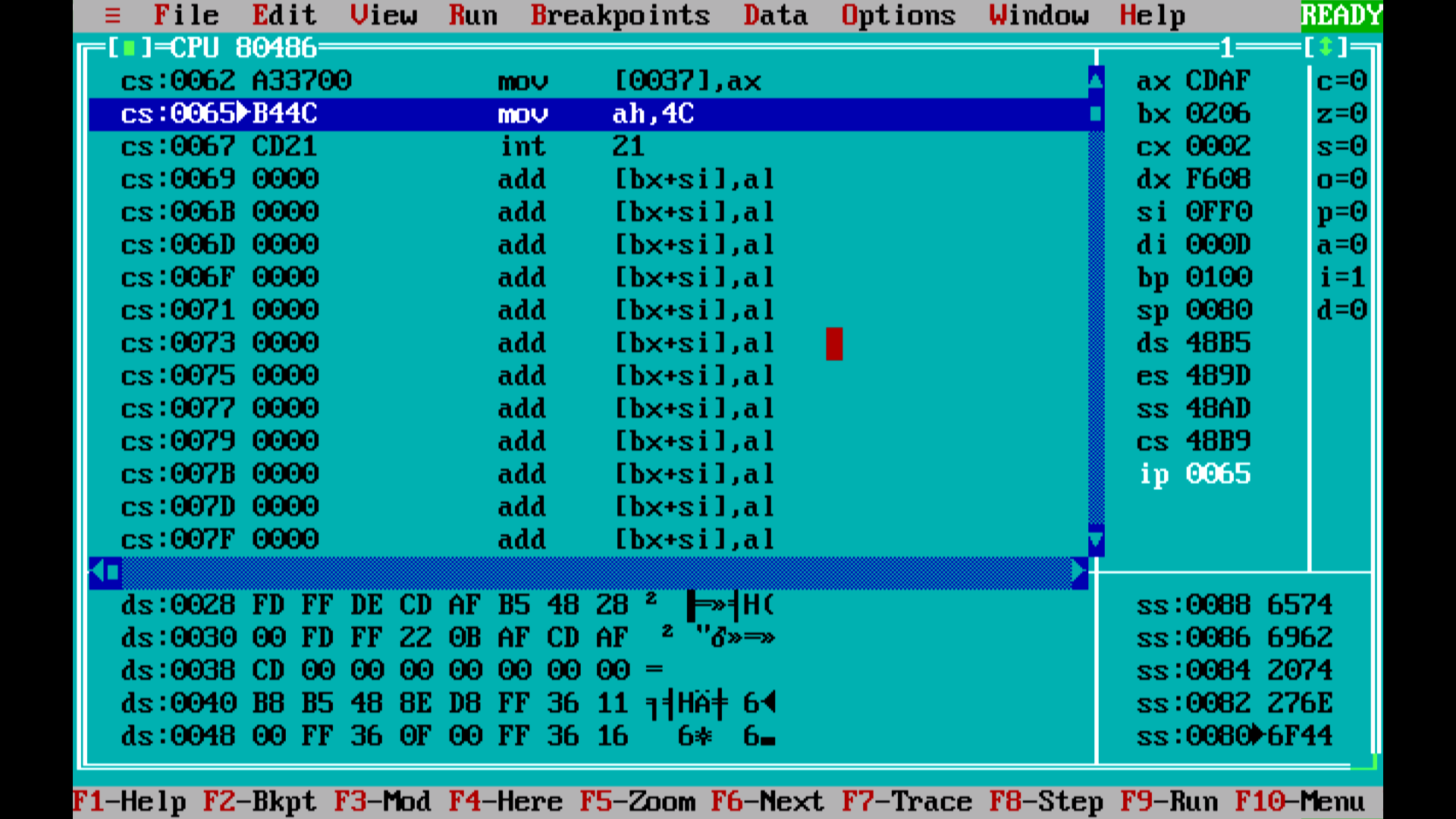
******

Рисунок 9

Програма працює корректно.

***4. Висновки щодо призначення регістрів, адресації даних, призначення команд пересилання й обміну та їх операндів.***

До стеку можна записувати лише 16бітні дані, зчитувати також тільки 16 біт.

Зчитуються тільки останні записані 16біт, тобто якщо ми запишемо слово1 слово2 слово3, то коли мі и будем їх зчитувати, ми зчитаемо слово3 слово2 слово1, тобто у зворотному порядку.

Якщо потрібно записати у змінну декілька різніх за розміром даних, наприклад слово, байт та подвійне слово, то треба записувати їх за допомогою команд x ptr [a], де x – назва директиви, яку треба записати, а – адреса зміщення відносно початку змінної, куди треба записати директиву.

Архітектура процессора 8086 має 6 типів регістрів процессора:

1)Загального призначення:

16ти розрядні: ax(accumulator), bx(base), cx(counter), dx(data)

Та 8ми розрядні, які входять у склад відповідних 16ти розрядних:

ah, al, bh, bl, ch, cl, dh, dl.

Ax використовуються для аріфметичних операцій, але також може використовуватись для зберігання змінних.

Bx по замовчуванню утримує адресу процедури або змінної, але також може використовуватись для аріфметичних операцій та зберігання змінних.

Cx використовуется як лічильник для циклу, але також може використовуватись для аріфметичних операцій та зберігання змінних.

Dx утримує дані для операцій вводу та виводу даних, але також може використовуватись для аріфметичних операцій та зберігання змінних.

2)Сегменти

Ss – сегмент стеку, використовуется для тимчасового зберігання даних, можна записувати/зчітувати тільки верхні 16 біт.

Ds – сегмент даних, зберігання змінних.

Cs – сегмент коду, код програми.

Es – додатковий сегмент, теж саме, що й ds.

3) Вказівники:

BP(base pointer) – зберігає зміщення від сегменту стєку.

Sp(stack pointer) – зберігає зміщення від сегменту стєку, разом із ss є адресою вершини стєку.

SI(source index) – використовуется у косвенній адресації та деяких цепочних операціях.

Di(destination index) – використовуется у цепочних операціях.

4)Контрольні:

IP(instruction pointer) – зберігає зміщення наступної команди сегменту коду. Разом із cs є повною адресою наступної команди

FR(flag register) – технічно їх 16(кожен по одному біту), але вікористовуется лише 9:

CF(Carry Flag) – використовуется для перевірки на переповнення для аріфметичних операций над беззнаковими числами.

PF(Parity Flag) – перевіряє кількисть одиниць у байті.

AF(Auxiliary Carry Flag) – перевірка на зміщення з молодшого до старшогу біту при додаванні.

ZF( Flag) – перевірка чи результат арифметичної операції дорівнює нулеві.

SF( Flag) – перевірка на знаковість числа.  
 OF(Overflow Flag) – перевірка на переповнення .

TF( Flag) – використовуется для відладки программи(виконуется одна команда за раз).

IF( Flag) – використовуется для зовнішніх преривань.

DF( Flag) – відповідає за направлення цепочних команд.

Існує декілька видів адрессації:

Прямий – вказуется відносна адреса від мітки або початку регістру та регістр, наприклад [F+x], де F – мітка або регістр, x – зміщення.

Регістрова – дані повністю займають регістр, тож вказуется тільки назва регістру.

Непряма – теж саме що й пряма але зміщення знаходится в одному з регістрів SI DI BX, наприклад DS:[DI].

Неявна – деякі команди працюють тільки з конкретним регістром, тож при їх використанні не вказується звідки брати дані, наприклад команда loop має лічильник у CX.