Программирование

Первая программа

Базовая архитектура IBM PC X86

Представление данных

Данные представляются в виде целых чисел в следующих форматах:

- 1 байт
- 2 байта = слово
- 4 байта = двойное слово
- 8 байтов = четверное слово

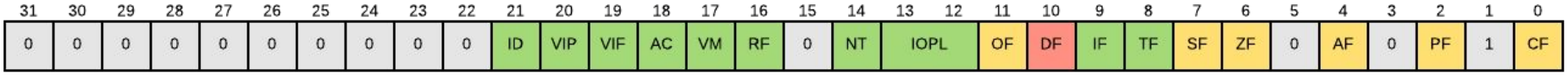
Кодирование данных:

- дополнительный код

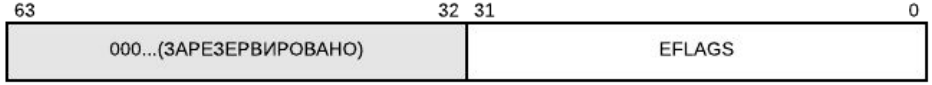
Базовая архитектура IBM PC X86

Регистры процессора: регистр флагов

EFLAGS



RFLAGS



FLAGS

В регистре хранятся данные о состоянии процессора и результатах выполнения некоторых команд.

Базовая архитектура IBM PC X86

Регистры процессора: регистр флагов

FLAGS

C – carry flag (флаг переноса) – выполнение операции привело к возникновению переноса

P – parity flag (флаг четности) – количество единиц в младшем байте результата чётно

A – auxiliary carry flag (флаг дополнительного переноса) – используется при операциях с двоично-десятичными числами

Z – zero flag (флаг нуля) – результатом операции был ноль

Базовая архитектура IBM PC X86

Регистры процессора: регистр флагов

FLAGS

S – sign flag (флаг знака) – старший разряд результата имеет значение «1»

T – trap flag (флаг трассировки) – используется программами-отладчиками

I – interrupt flag (флаг прерывания) – процессор реагирует на прерывания

D – direction flag (флаг направления) – используется командами обработки строк

O – overflow flag (флаг переполнения) – устанавливается при переполнении (результат операции не помещается в регистре

Базовая архитектура IBM PC X86

Регистры процессора: РОНы

Базовая архитектура IBM PC X86

Регистры процессора: РОНы

Регистр AX (accumulator, аккумулятор)

Это регистр-накопитель. Наиболее эффективно его использование в арифметических и логических операциях, а также в операциях пересылки, т.к. именно эти операции оптимизированы для использования регистра AX и, как правило, обладают более высоким быстродействием.

Базовая архитектура IBM PC X86

Регистры процессора: РОНы

Регистр BX (base, базовый регистр)

В некоторых операциях этот регистр используется для реализации расширенной адресации.

Базовая архитектура IBM PC X86

Регистры процессора: РОНы

Регистр CX (counter, счётчик)

Обычно этот регистр используется как счётчик, указывающий количество выполнений команды или группы команд (циклические вычисления, сдвиги).

Базовая архитектура IBM PC X86

Регистры процессора: РОНы

Регистр DX (data, регистр данных)

Этот регистр используется в операциях умножения и деления, а также является единственным регистром, в котором может быть указан адрес порта в командах ввода-вывода.

Базовая архитектура IBM PC X86

Регистры процессора: РОНы

Регистр IP (instruction pointer, счётчик команд)

Регистр содержит адрес команды, следующей за выполняемой в текущий момент, в сегменте памяти, который задаётся регистром CS.

Ассемблер Intel 8086

Команды пересылки данных: mov

- **Общий формат**: mov Operand1, Operand2
- **Действие**: Operand1 = Operand2
- **Описание**: команда предназначена для передачи значения второго операнда первому. В зависимости от описания операндов пересылается слово или байт. Если операнды описаны по-разному или нельзя однозначно определить размер операнда, используется один из атрибутных операторов: **byte ptr** или **word ptr.**
- **Особенность**: запрещены пересылки из ячейки памяти в ячейку памяти.
- **Примеры**:

```
mov AX, BX
mov AH, Mem_DB
mov Mem_DW, CX
mov DS, AX
mov DX, ES
mov Value_SS, SS
mov CH, 200
mov Mem_DB, 200
mov word ptr [bx], 1  ; mov byte ptr [bx], 1
```

Ассемблер Intel 8086

Арифметические операции: add, adc

- **Общий формат**: add Operand1, Operand2
- **Действие**: Operand1 := Operand1 + Operand2
- **Описание**: сложение двух целых чисел со знаком или без знака.
- **Особенность**: Operand1 и Operand2 не могут быть ячейками памяти одновременно.

- **Общий формат**: adc Operand1, Operand2
- **Действие**: Operand1 := Operand1 + Operand2 + CF.
- **Описание**: сложение двух целых чисел со знаком или без знака, при этом в операции принимает участие и флаг переноса из регистра флагов.
- **Особенность**: Operand1 и Operand2 не могут быть ячейками памяти одновременно.

- **Пример**: сложение двух 32-разрядных чисел (операнд1 – BX:AX, операнд2 – DX:CX)
- Add AX, CX
- Adc BX, DX

Ассемблер Intel 8086

Арифметические операции: sub, sbb

- **Общий формат**: sub Operand1, Operand2
- **Действие**: Operand1 := Operand1 – Operand2
- **Описание**: вычитание двух целых чисел со знаком или без знака.
- **Особенность**: Operand1 и Operand2 не могут быть ячейками памяти одновременно.

- **Общий формат**: sbb Operand1, Operand2
- **Действие**: Operand1 := Operand1 – Operand2 – CF.
- **Описание**: вычитание двух целых чисел со знаком или без знака, при этом в операции принимает участие и флаг переноса из регистра флагов.
- **Особенность**: Operand1 и Operand2 не могут быть ячейками памяти одновременно.

Ассемблер Intel 8086

Арифметические операции: mul, imul

- **Общий формат**: mul Operand
- **Описание**: умножение двух целых чисел без знака. Первый операнд хранится в регистре AL (при умножении байтов) или AX (при умножении слов), второй операнд задаётся в команде. Результат помещается в регистр AX или в регистры DX:AX соответственно.
- **Особенность**: разрядность операции задаётся разрядностью Operand.

- **Общий формат**: imul Operand
- **Описание**: умножение двух целых чисел со знаком. Первый операнд хранится в регистре AL (при умножении байтов) или AX (при умножении слов), второй операнд задаётся в команде. Результат помещается в регистр AX или в регистры DX:AX соответственно.
- **Особенность**: разрядность операции задаётся разрядностью Operand.

- **Пример**:

```
Mov AX, 100h
Mul AX          ; = AX*AX, результат – DX:AX
```

Ассемблер Intel 8086

Арифметические операции: div, idiv

- **Общий формат**: div Operand
- **Описание**: деление двух целых чисел без знака.
- **Особенность**: разрядность операции задаётся разрядностью делителя – Operand.

- **Общий формат**: idiv Operand
- **Описание**: деление двух целых чисел со знаком.
- **Особенность**: разрядность операции задаётся разрядностью делителя – Operand.

Делимое	Делитель	Частное	Остаток
AX	8 разрядов	AL	AH
DX:AX	16 разрядов	AX	DX

Ассемблер Intel 8086

Арифметические операции: cmp

- **Общий формат**: cmp Operand1, Operand2
- **Описание**: выполняется сравнение двух операндов путём вычитания второго операнда из первого. Результат операции не записывается, вместо этого устанавливаются значения флагов в регистре флагов процессора.

Ассемблер Intel 8086

Команды передачи управления: jmp

- **Общий формат**: jmp Target
- **Описание**: выполняется передача управления по адресу, заданному параметром команды Target. Адрес может задаваться как напрямую (с помощью метки), так и с помощью регистров и ячеек памяти.
- **Особенность**: переход внутри одного сегмента задаётся только смещением, переход между сегментами задаётся полным адресом.
- **Виды безусловных переходов**:
- 1) прямой короткий (пример: jmp short Point1);
- 2) прямой ближний (пример: jmp near ptr Point2);
- 3) прямой дальний (пример: jmp far ptr Point3);
- 4) косвенный ближний (пример: jmp word ptr [SI+2]);
- 5) косвенный дальний (пример: jmp dword ptr [DX]).

- **Другие примеры**:
- jmp BX
- jmp SkipAdd
- jmp dword ptr AddrTable[SI+2]

Ассемблер Intel 8086

Команды передачи управления: j?*

- **Общий формат**: j?* Target
- **Описание**: при выполнении некоторого условия выполняется передача управления по адресу, заданному параметром команды Target. Адрес может задаваться как напрямую (с помощью метки), так и с помощью регистров и ячеек памяти.
- **Особенность**: выполняется только короткий переход, т.е. адрес, заданный операндом Target, должен быть расположен в диапазоне от -128 до +127 байтов от адреса, хранимого в регистре IP.

- При написании команд удобно пользоваться сокращениями:

A – above (выше)	**B** – below (ниже)
C – carry (перенос)	**E** – equal (равно)
G – greater (больше)	**L** – less (меньше)
N – not (не)	**O** – overflow (переполнение)
P – parity (паритет)	**S** – sign (знак)
Z – zero (ноль)	

Ассемблер Intel 8086

Команды передачи управления: j?*

- **Примеры конструирования команд условного перехода:**
- **ja, jnbe** – переход, если выше/переход, если не ниже и не равно
- **jg, jnle** – переход, если больше/переход, если не меньше и не равно
- **jc** – переход, если установлен флаг переноса
- **jz, je** – переход, если ноль/переход, если равно
- …

- **Дополнительная команда условного перехода:**
- **jcxz** – переход, если CX=0

Ассемблер Intel 8086

Команды передачи управления: loop

- **Общий формат**: loop LoopLabel
- **Описание**: команда предназначена для организации циклических вычислений. Количество повторений задаётся в регистре CX. Выход из цикла происходит, когда при очередной проверке CX оказывается равным нулю.
- **Алгоритм**:
- CX := CX – 1;
- if CX <> 0 then jmp LoopLabel;
- **Особенность**: выполняется только короткий переход, т.е. адрес, заданный операндом LoopLabel, должен быть расположен в диапазоне от -128 до +127 байтов от адреса, хранимого в регистре IP.

- **Пример**:
- mov AX, 0
- mov CX, 100
- AXIncLoop:
- add AX, 1
- loop AXIncLoop

Ассемблер Intel 8086

Команды передачи управления: loope, loopz

- **Общий формат**: loope LoopLabel
- **Описание**: команда предназначена для организации циклических вычислений. Количество повторений задаётся в регистре CX. Кроме значения регистра CX данная команда анализирует содержимое флага Z регистра флагов. Цикл выполняется, пока содержимое регистра CX не равно нулю и флаг ZF равен 1.
- **Алгоритм**:
- CX := CX – 1;
- if (CX <> 0) and (ZF = 1) then jmp LoopLabel;
- **Особенность**: выполняется только короткий переход, т.е. адрес, заданный операндом LoopLabel, должен быть расположен в диапазоне от -128 до +127 байтов от адреса, хранимого в регистре IP.

- loopz – синоним команды loope.

Ассемблер Intel 8086

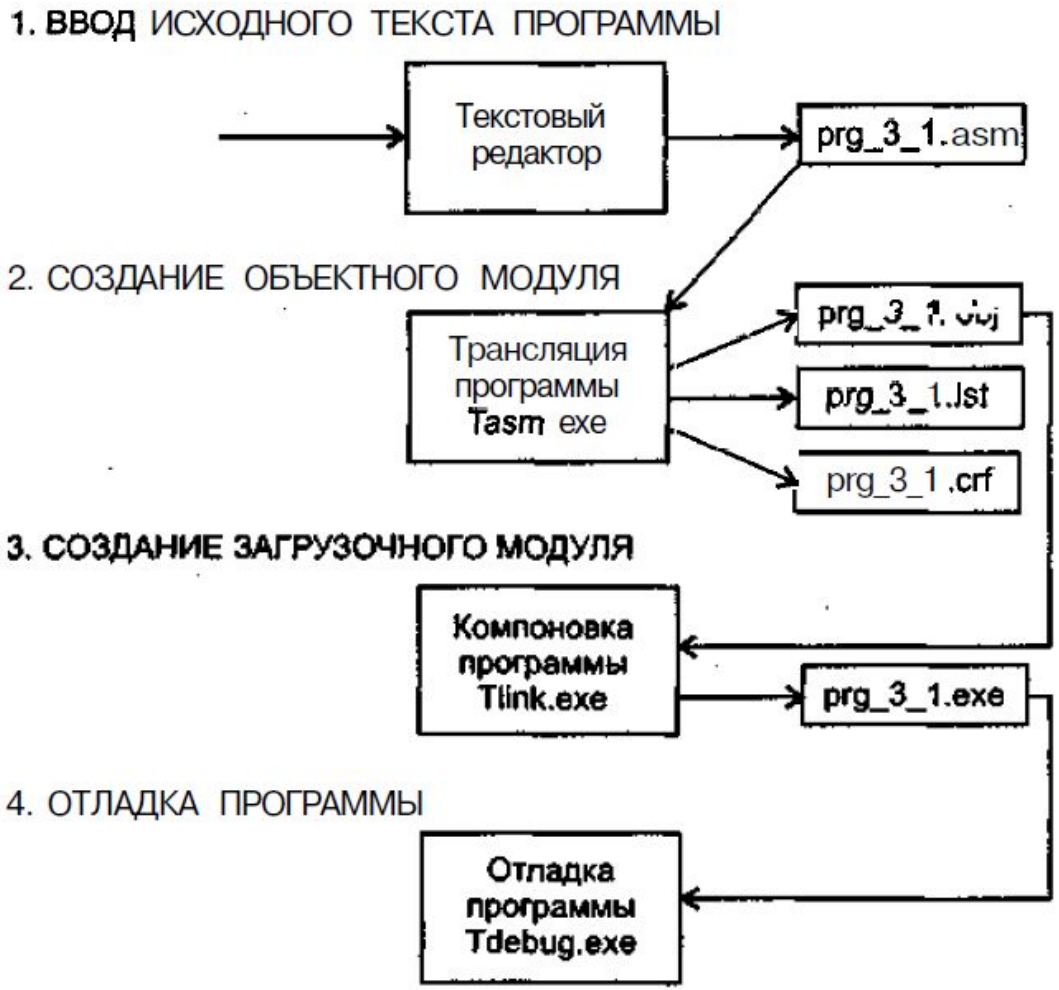
Команды передачи управления: loopne, loopnz

- **Общий формат**: loopne LoopLabel
- **Описание**: команда предназначена для организации циклических вычислений. Количество повторений задаётся в регистре CX. Кроме значения регистра CX данная команда анализирует содержимое флага Z регистра флагов. Цикл выполняется, пока содержимое регистра CX не равно нулю и флаг ZF равен нулю.
- **Алгоритм**:
- CX := CX – 1;
- if (CX <> 0) and (ZF = 0) then jmp LoopLabel;
- **Особенность**: выполняется только короткий переход, т.е. адрес, заданный операндом LoopLabel, должен быть расположен в диапазоне от -128 до +127 байтов от адреса, хранимого в регистре IP.

- loopnz – синоним команды loopne.

Ассемблер Intel 8086

Схема процесса разработки программы на ассемблере

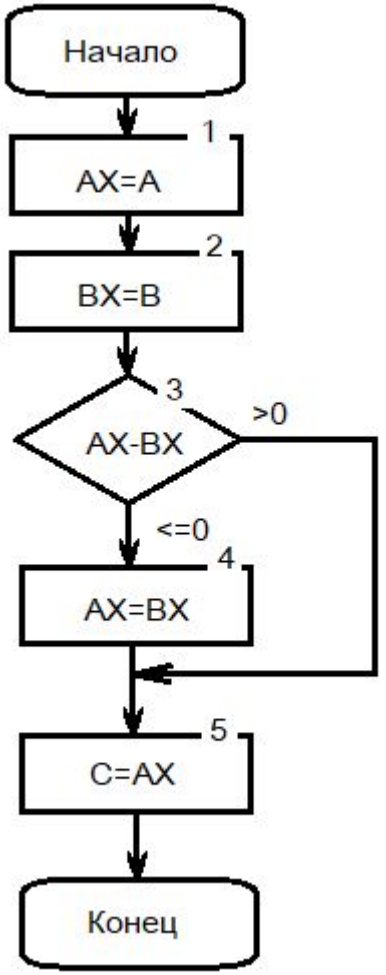


Ассемблер Intel 8086

Создание программы в текстовом редакторе

Даны два числа (A, B). Переменной C присвоить большее значение из A и B.

```
.model tiny
.data
A    dw  7
B    dw  5
C    dw  ?
Start:
     mov     ax,@Data
     mov     ds, ax
     mov ax, A   ;блок 1
     mov bx, B   ;блок 2
     cmp ax, bx  ;блок 3
     jg  great
     mov ax, bx  ;блок 4
great:
     mov C, ax   ;блок 5
     mov ax, 4C00h
     int 21h
     end Start
```



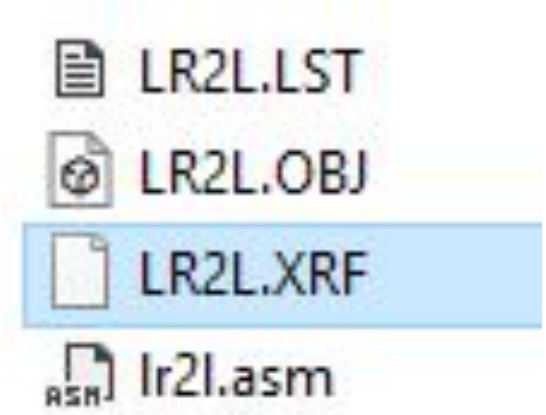
Ассемблер Intel 8086

Трансляция программы

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DOSBOX

C:\>tasm lr2l,,,
Turbo Assembler  Version 1.0  Copyright (c) 1988 by Borland International

Assembling file:   LR2L.ASM
Error messages:    None
Warning messages:  None
Remaining memory:  478k
```



Ассемблер Intel 8086

Трансляция программы

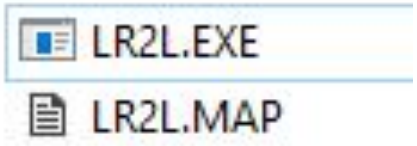
```
Turbo Assembler  Version 1.0        02-09-22 01:50:14
Page 1
LR2L.ASM

      1 0000                        .model tiny
      2 0000                        .data
      3
      4 0000  0007                  A       dw      7
      5 0002  0005                  B       dw      5
      6 0004  ????                  C       dw      ?
      7
      8 0006                        Start:
      9 0006  B8 0000s                      mov     ax,@Data
     10 0009  8E D8                         mov     ds, ax
     11 000B  A1 0000r                      mov     ax, A
     12 000E  8B 1E 0002r                   mov     bx, B
     13 0012  3B C3                         cmp     ax, bx
     14 0014  7F 02                         jg      great
     15 0016  8B C3                         mov     ax, bx
     16 0018                        great:
     17 0018  A3 0004r                      mov     C, ax
     18 001B  B8 4C00                       mov     ax, 4C00h
     19 001E  CD 21                         int     21h
     20
     21                                     end     Start
```

Ассемблер Intel 8086

Трансляция программы

```
C:\>tlink lr2l
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International
Warning: No stack
```



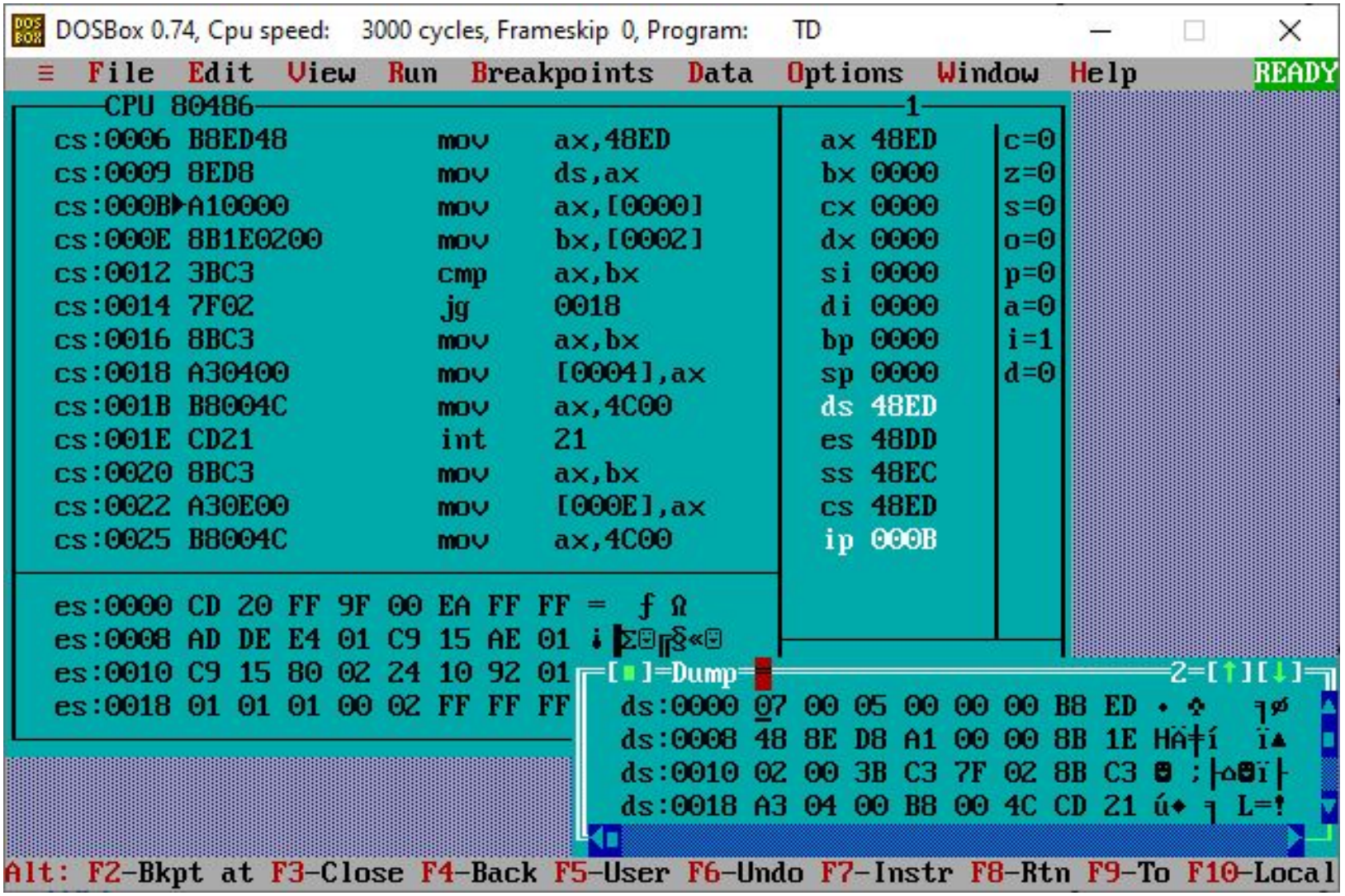

```
 Start  Stop   Length Name                    Class

 00000H 00000H 00000H _TEXT                   CODE
 00000H 0001FH 00020H _DATA                   DATA

Program entry point at 0000:0006
Warning: No stack
```

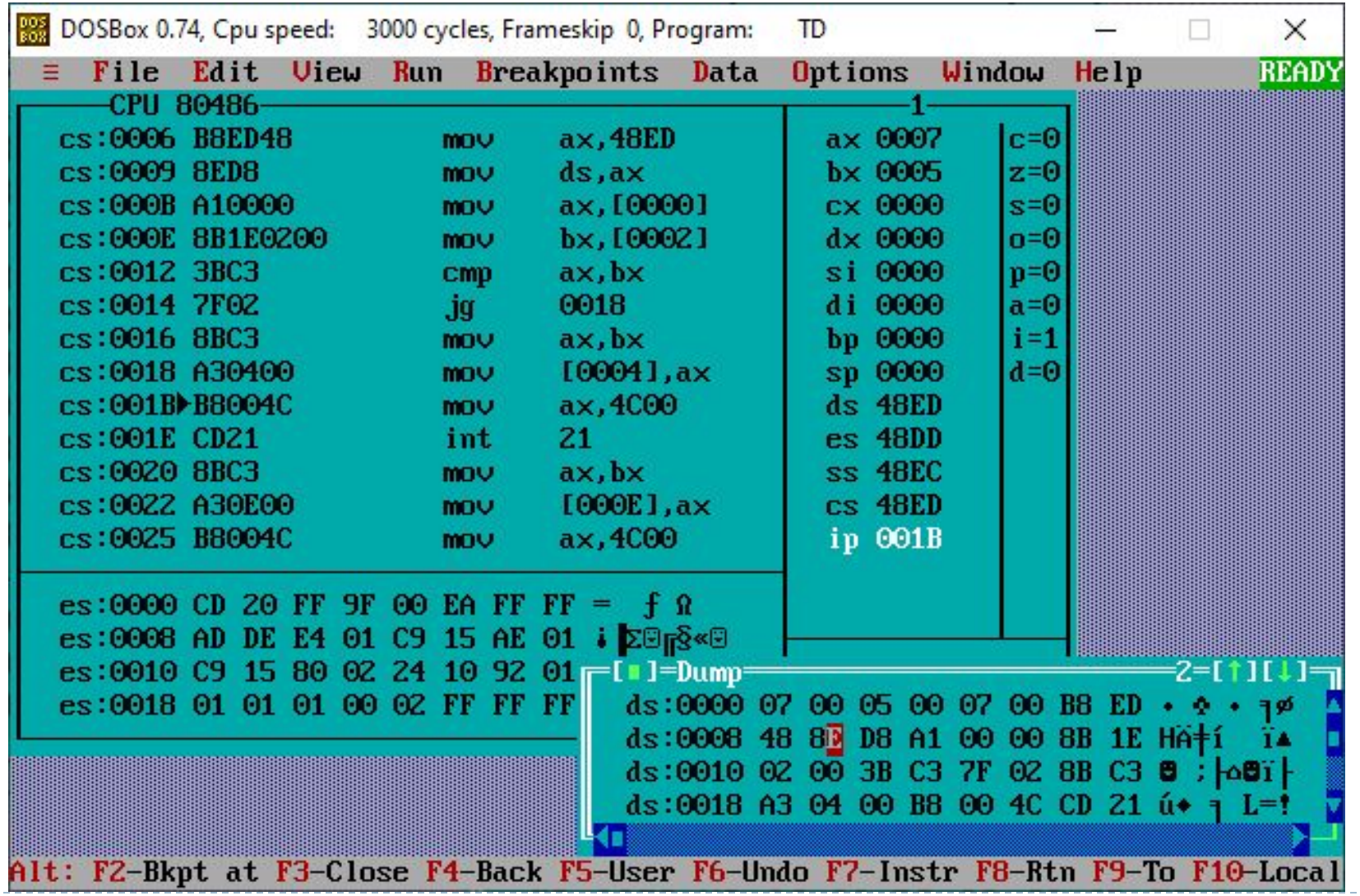
Ассемблер Intel 8086

Отладка программы



Ассемблер Intel 8086

Отладка программы



Ассемблер Intel 8086

Отладка программы