

Весна 2022

Системное программное обеспечение

Онлайн-лекции

Лекция №3: **Отладчик кодов**

Доцент, к.т.н. ГОЛЬЦОВ Александр Геннадьевич

Отладчик кодов (debugger)

Это инструментальная программа, позволяющая:

- загружать исполнимые файлы, эмулируя действия загрузчика из состава операционной системы;
- дизассемблировать код (расшифровывать мнемоники машинных команд по имеющимся в памяти машинным кодам);
- просматривать содержимое регистров процессора и областей памяти;
- исполнять программу целиком и по шагам (машинным командам);
- устанавливать точки останова (**breakpoint**) при выполнении программы;
- изменять области памяти и значения в регистрах;
- ассемблировать машинные команды (переводить мнемоники в машинный код)
- и др.

Символьный отладчик

Это инструментальная программа, позволяющая:

- отлаживать исполнимые файлы, написанные на том или ином **языке программирования**;
- использовать **имена** меток, процедур и переменных, заданные программистом - разработчиком отлаживаемой программы;
- контролировать значения переменных;
- исполнять операторы программы на языке программирования по шагам и программу целиком;
- устанавливать точки останова при выполнении программы;
- изменять значения переменных;
- обычно входят в состав интегрированных сред разработки (IDE) на ЯВУ (Си, Паскаль-Дельфи, FoxPro, ...) и часто работают прямо в окне редактора исходного текста.

Особенности работы программы под⁴ отладчиком

- Программа выполняется непосредственно процессором с реальной скоростью процессора, отладчик не эмулирует работу процессора и, как правило, не замедляет программу
- Отладчик тесно взаимодействует с аппаратурой ЭВМ, активно внедряется в систему прерываний

Примеры отладчиков (DOS, i8086) ⁵

- **Debug** - отладчик кодов, составная часть операционной системы DOS, интерфейс командной строки
- **AFD** - отладчик кодов, интерактивная работа
- **Turbo Debugger**:
 - отладчик кодов
 - **символьный** отладчик для программ на Си, Паскале и языке ассемблера
 - входил в состав пакетов Turbo/Borland C/Pascal

Загрузка программы

- Командная строка:
td my.exe
- Меню:
File → Open... → имя exe-файла или com-файла
- После загрузки память выделена и регистры процессора проинициализированы так, как сделал бы загрузчик DOS
- **Ctrl+F2** - сброс программы, вернуть все в состояние как будто сразу после загрузки.

Отлаживаем все тот же пример:

```
.model small      ; один сегмент кода, данных и стека
.stack 100h       ; отвести под стек 256 байт
.data            ; начало сегмента данных
S      db      'Hello, world!$'
.code           ; начало сегмента кода
; Начальная инициализация
mov  ax,@data
mov  ds,ax      ; настройка DS на начало сегмента данных

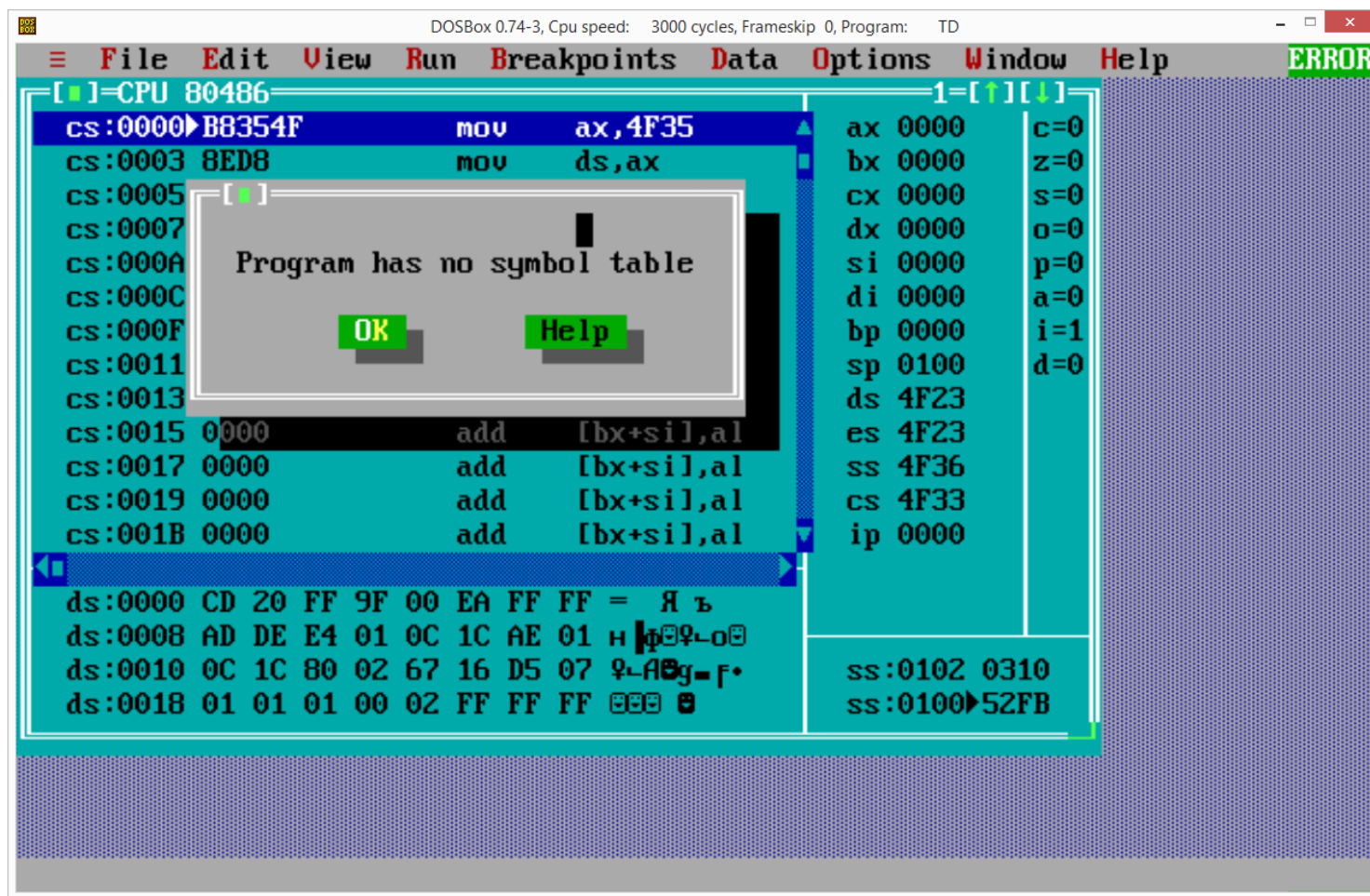
;-----
; Здесь - код в соответствии с заданием, например
; Вывод строки на экран
mov  ah,9       ; номер функции DOS
mov  dx,offset S; DS:DX <- адрес строки S
                ; DS уже проинициализирован ранее
int  21h        ; Вывод строки на экран в текущей позиции курсора
;-----

; Стандартное завершение программы
mov  ax,4C00h   ; ah = N функции, al = код возврата
int  21h       ; снять программу с выполнения

end           ; конец текста программы
```

Окно ЦПУ

- Меню: View → CPU




```

≡ File Edit View Run Breakpoints Data Options Window Help READY
[ ]=CPU 80486
cs:0000 B8354F mov ax,4F35
cs:0003 8ED8 mov ds,ax
cs:0005 B409 mov ah,09
cs:0007 BA0000 mov dx,0000
cs:000A CD21 int 21
cs:000C B8004C mov ax,4C00
cs:000F CD21 int 21
cs:0011 0000 add [bx+si],al
cs:0013 0000 add [bx+si],al
cs:0015 0000 add [bx+si],al
cs:0017 0000 add [bx+si],al
cs:0019 0000 add [bx+si],al
cs:001B 0000 add [bx+si],al
cs:001D 0000 add [bx+si],al
cs:001F 004865 add [bx+si+65],cl

ds:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
ds:0008 AD DE E4 01 0C 1C AE 01 H 00 00 00
ds:0010 0C 1C 80 02 67 16 D5 07 00 00 00
ds:0018 01 01 01 00 02 FF FF FF 00 00 00
ds:0020 FF FF FF FF FF FF FF FF

ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0100 d=0
ds 4F23
es 4F23
ss 4F36
cs 4F33
ip 0000

ss:0102 0310
ss:0100 52FB
ss:00FE FFFF
ss:00FC 0000
ss:00FA 0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

[CPU 80486] ds:519B = 00 3

cs:0000 B8354F	mov	ax,4F35	ax	0935	c=0
cs:0003 8ED8	mov	ds,ax	bx	4F37	z=0
cs:0005 B409	mov	ah,09	cx	0405	s=0
cs:0007 BA0000	mov	dx,0000	dx	0000	o=0
cs:000A CD21	int	21	si	0264	p=0
cs:000C B8004C	mov	ax,4C00	di	0150	a=0
cs:000F CD21	int	21	bp	3FF4	i=1
cs:0011 0000	add	[bx+si],al	sp	0100	d=0
cs:0013 0000	add	[bx+si],al	ds	4F35	
cs:0015 0000	add	[bx+si],al	es	4F23	
cs:0017 0000	add	[bx+si],al	ss	4F36	
cs:0019 0000	add	[bx+si],al	cs	4F33	
cs:001B 0000	add	[bx+si],al	ip	000A	
cs:001D 0000	add	[bx+si],al			
cs:001F 004865	add	[bx+si+65],cl			

es:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
 es:0008 AD DE E4 01 0C 1C AE 01 H ф ф ф ф ф ф
 es:0010 0C 1C 80 02 67 16 D5 07 ф ф ф ф ф ф
 es:0018 01 01 01 00 02 FF FF FF ф ф ф ф ф ф
 es:0020 FF FF FF FF FF FF FF FF

ss:0108 0003
 ss:0106 0000
 ss:0104 000C
 ss:0102 0310
 ss:0100 52FB

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local

Текущая команда - треугольник (CS:IP). Измененные регистры - белые.
 Ячейка, используемая подсвеченной командой - вверху на рамке.

Употребительные клавиши

- **F5** - распаковать окно отладчика
- **F10** - перейти в меню
- **Alt+F10** - вызвать контекстное меню (правая кнопка мыши)
- **Alt+F5** - посмотреть вид экрана, на который программа производит вывод
- Клавиши выводятся в нижней строке
- При нажатии **Alt** и **Ctrl** - информация меняется

Прокрутка кода клавишами

- Текущая команда отмечена треугольником
- Прокручивать окно дизассемблера вниз можно всегда
- При попытке прокрутить окно вверх дизассемблер очень часто ошибается, хватаясь за самый длинный предшествующий машинный код:

cs:0000 ►B8354F	mov	ax,4F35	cs:FFFF 00B8354F	add	[bx+si+4F35],bh
cs:0003 8ED8	mov	ds,ax	cs:0003 8ED8	mov	ds,ax
cs:0005 B409	mov	ah,09	cs:0005 B409	mov	ah,09
cs:0007 BA0000	mov	dx,0000	cs:0007 BA0000	mov	dx,0000
cs:000A CD21	int	21	cs:000A CD21	int	21
cs:000C B8004C	mov	ax,4C00	cs:000C B8004C	mov	ax,4C00
cs:000F CD21	int	21	cs:000F CD21	int	21

обратите внимание: cs:FFFF=00 - это не предыдущий байт перед cs:0000, это самый конец того же сегмента, смещение которого на 64к БОЛЬШЕ смещения 0

Позиционирование

- Контекстное меню: Goto
 - просто число → смещение в том же сегменте
 - два числа через : → сегмент и смещение
 - имя сегментного регистра:число → сегмент и смещение
 - можно указать любое корректное выражение в качестве адреса, например `ds:si` или `ax`
 - просто `ds` → смещение, равное DS!
- При вводе чисел не забываем правила языка ассемблера:
 - число начинается с цифры
 - B, D или H в конце - явный признак формата (двоичное, десятичное, шестнадцатеричное)

Исполнение программы

- **F9** - исполнить с текущей команды до конца (до команды завершения программы) или до точки останова
- **F7** - шаг: исполнить текущую команду и остановиться после этого (если исполняется вызов процедуры – войти в процедуру)
- **F8** - шаг: исполнить текущую и остановиться перед следующей командой (если вызов процедуры – выполнить ее целиком и вернуться)
- **F4** - "сюда": исполнять программу с текущей команды и остановиться на подсвеченной строке
- **F2** - установить/снять точку останова на подсвеченной строке (ТО выделяются красным)

Точки останова (ловушки?)

- **Локальные и глобальные:** при выполнении конкретной команды или при выполнении условия
- **Безусловные:** дошли до конкретной команды - остановились
- **Условные:** выполнилось условие - остановились (**замедленное выполнение** при наличии таких точек останова) → локальные и глобальные !
- Условия:
 - произошло обращение к **области памяти** по определенному адресу известного размера
 - выполнилось условие, например, **$DX \text{ eq } 8$**
- Точки останова можно **объединять в группы**, чтобы разрешать/запрещать сразу несколько

Как работают точки останова?

- Если включен режим отладки (флаг TF) - после каждой машинной команды генерируется прерывание отладки (прерывание с номером 1) - при его обработке у отладчика есть возможность вычислить глобальное условие
- Команда INT 3 (машинный код CCh): вызов прерывания отладчика
- Отладчик заменяет этой командой места в коде, где нужно остановиться; при срабатывании прерывания отладчик восстанавливает код "как было"

cs:0005	B409	mov	ah,09
cs:0007	BA0000	mov	dx,0000
cs:000A	CD21	int	21



B409
CC0000
CD21

Как и что можно изменять?

- Регистр. Контекстное меню: increment, decrement, change
- Флаг. Контекстное меню: toggle
- Ячейка стека. Контекстное меню: change
- Область дампа.
 - Контекстное меню: change
 - Или просто начать набирать в области дампа или в области символьного вида
- Код. Контекстное меню: assemble

Как искать?

- В области дампа. Контекстное меню: search
 - ввести строку в кавычках или значения байтов
- В области кода. Контекстное меню: search
 - ввести машинный код (байты): **0CDh**
 - или ввести мнемонику команды: **int 21**

Операции с блоками памяти

- В окно дампа, настроиться на нужную область памяти
- Контекстное меню: **Block**:
 - Clear - залить нулями
 - Move - переместить
 - Set - залить указанным значением
 - Read - прочитать содержимое из файла
 - Write - записать содержимое в файл
- Блок можно выделить в отладчике мышью или указывать его адрес и размер цифрами

Символьная отладка

- Turbo Debugger - **символьный** отладчик
- Если в скомпилированную программу включить отладочную информацию - он сможет шагать не по машинным командам, а по строкам вашего исходного текста, и показывать не адреса, а имена меток и переменных
- Он понимает не только ассемблерный исходник, но и исходник на Турбо-Си или Турбо-Паскале
- **Исходный текст и исполнимый файл нужно разместить рядом**, имя файла исходного текста прописано в отладочной информации в EXE

Исходники ASM

tasm -zi my.asm

tlink -v my.obj

td my

Вернуться в это окно - **F3** или View: Module

```

File Edit View Run Breakpoints Data Options Window Help
[ ] Module: my File: my.asm 11
; комментарий с указанием:
; - фамилии и группы студента
; - варианта
; - краткой формулировки задания
.model small ; один сегмент кода, данных и стека
.stack 100h ; отвести под стек 256 байт
.data ; начало сегмента данных
S db 'Hello, world!$'
.code ; начало сегмента кода
; Начальная инициализация
mov ax,@data
mov ds,ax ; настройка DS на начало сегмента данных
;-----
; Здесь ? код в соответствии с заданием, например
; Вывод строки на экран
mov ah,9 ; номер функции DOS
mov dx,offset S ; DS:DX <- адрес строки S
  
```

Watches

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local

Переходим в окно CPU

```

cs:0000 B8354F  ♦ mov ax,@data
cs:0003 8ED8      ♦ mov ds,ax ; настройка DS на начало
cs:0005 B409      ♦ mov ah,9 ; номер функции DOS
cs:0007 BA0000    ♦ mov dx,offset S ; DS:DX ← адрес ст
cs:000A CD21      ♦ int 21h ; Вывод строки на экран в т
cs:000C B8004C    ♦ mov ax,4C00h ; ah = N функции, al =
cs:000F CD21      ♦ int 21h ; снять программу с выполне
cs:0011 0000      add [bx+si],al
cs:0013 0000      add [bx+si],al
cs:0015 0000      add [bx+si],al
cs:0017 0000      add [bx+si],al
cs:0019 0000      add [bx+si],al
cs:001B 0000      add [bx+si],al
cs:001D 0000      add [bx+si],al
cs:001F 004865    add [bx+si+65],cl

```

```

CPU 80486
#my#11: mov ax,@data
cs:0000 B8354F    mov ax,4F35
#my#12: mov ds,ax ; настройка DS на нача
cs:0003 8ED8      mov ds,ax
#my#17: mov ah,9 ; номер функции DOS
cs:0005 B409      mov ah,09
#my#18: mov dx,offset S ; DS:DX ← адрес
cs:0007 BA0000    mov dx,0000
#my#20: int 21h ; Вывод строки на экран
cs:000A CD21      int 21
#my#24: mov ax,4C00h ; ah = N функции, a
cs:000C B8004C    mov ax,4C00
#my#25: int 21h ; снять программу с выпо
cs:000F CD21      int 21
cs:0011 0000      add [bx+si],a

```

Контекстное меню: **Mixed**:

- **Both** - отображать исходный текст вместо дизассемблера, если возможно
- **No** - отображать только результат дизассемблирования
- **Yes** - отображать исходный текст и машинный код, получающийся из него (это особенно интересно для ЯВУ)

Goto...
 Origin
 Follow
 Caller
 Previous
 Search...
 View source
Mixed Yes
 New cs:ip
 Assemble...
 I/O

Используем имя метки

- Переходим в окно дампа
- Goto в контекстном меню
- Вводим просто **s** (имя метки данных строки)

```

cs:0013 0000          add     [bx+si],al
cs:0015 00          [ ]=Enter address to position to
cs:0017 00
cs:0019 00          s
cs:001B 00
cs:001D 00          OK  Clip...  Cancel  Help
cs:001F 00

ds:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
ds:0008 AD DE E4 01 0C 1C AE 01 н фф-оф
ds:0010 0C 1C 80 02 67 16 D5 07 ф-Афг-г•
ds:0018 01 01 01 00 02 FF FF FF ффф ф
ds:0020 FF FF FF FF FF FF FF FF

```



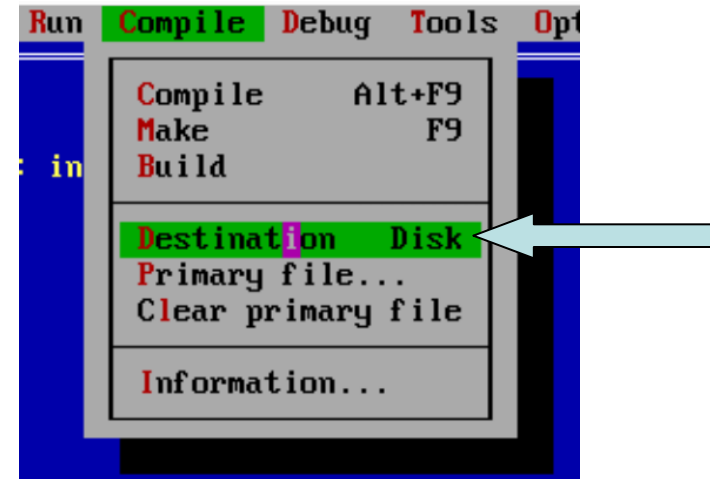
```

4F35:0000 48 65 6C 6C 6F 2C 20 77 Hello, w
4F35:0008 6F 72 6C 64 21 24 00 00 orld!$
4F35:0010 00 00 00 00 00 00 00 00
4F35:0018 00 00 00 00 00 00 00 00
4F35:0020 00 00 00 00 00 00 00 00

```

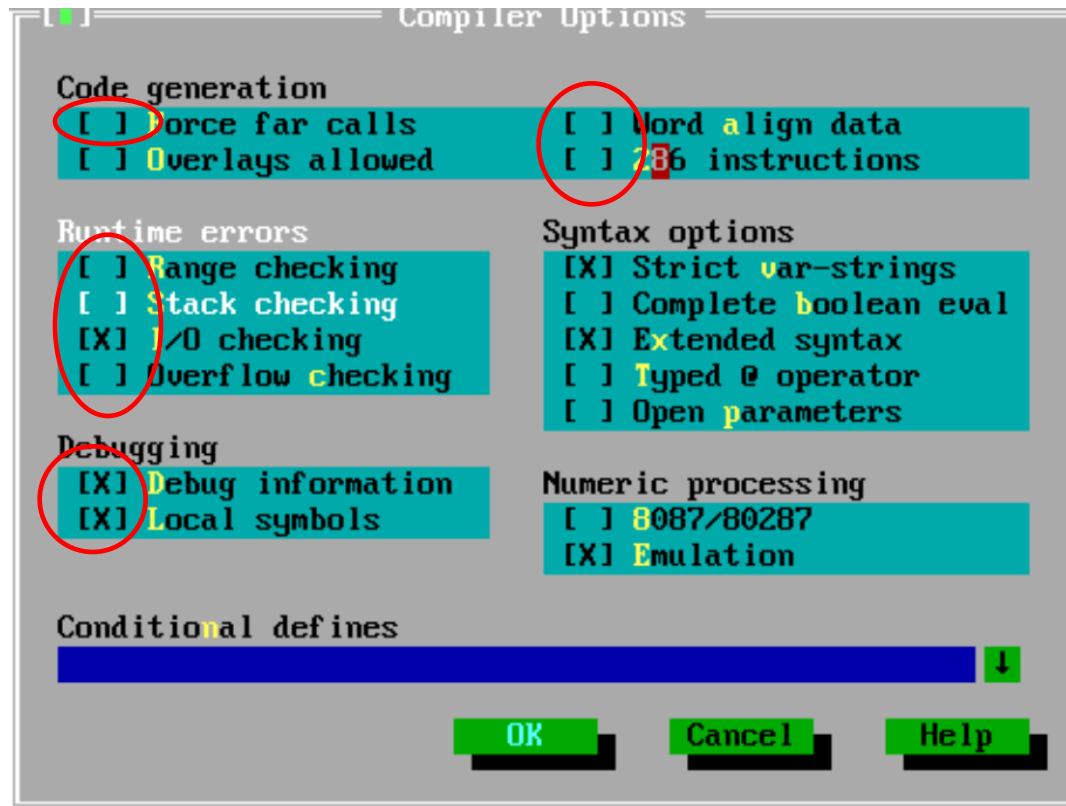
Программа на ЯВУ (Паскаль)²⁴

```
program MyPas;  
  
procedure Calc(a,b,c: integer);  
  var x,y: integer;  
begin  
  x:=a+b;  
  y:=c-x;  
end;  
  
begin  
  Calc(5,10,15);  
end.
```



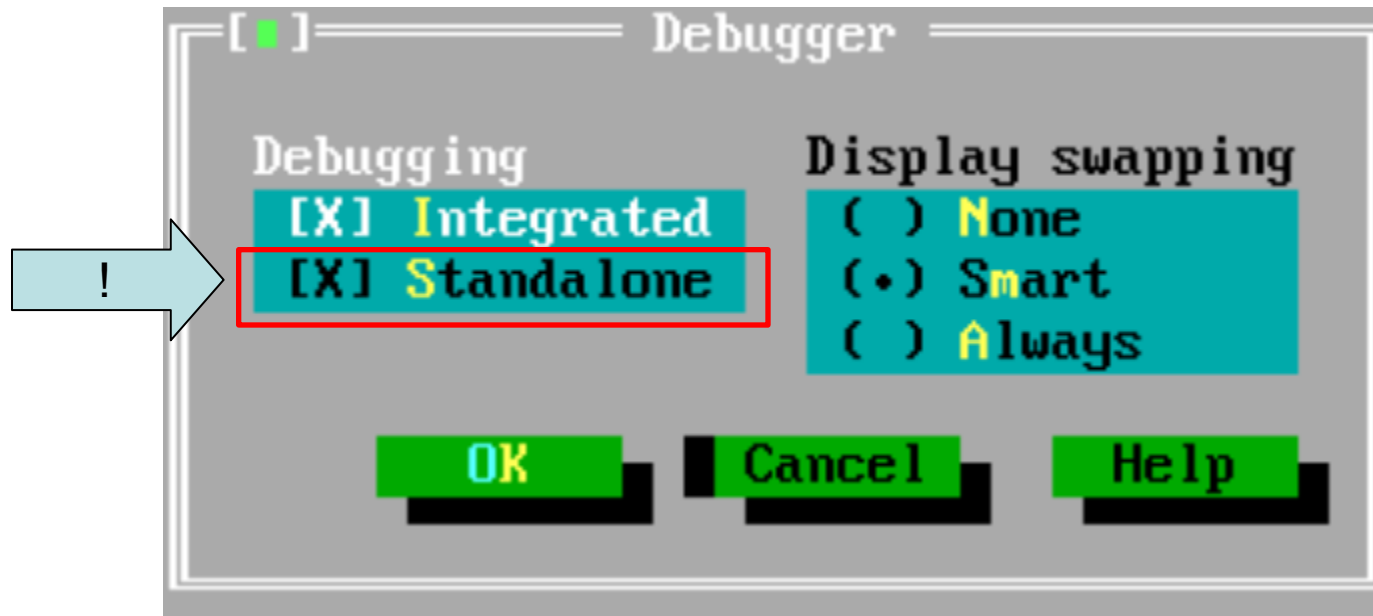
Опции компилятора

Options → Compiler



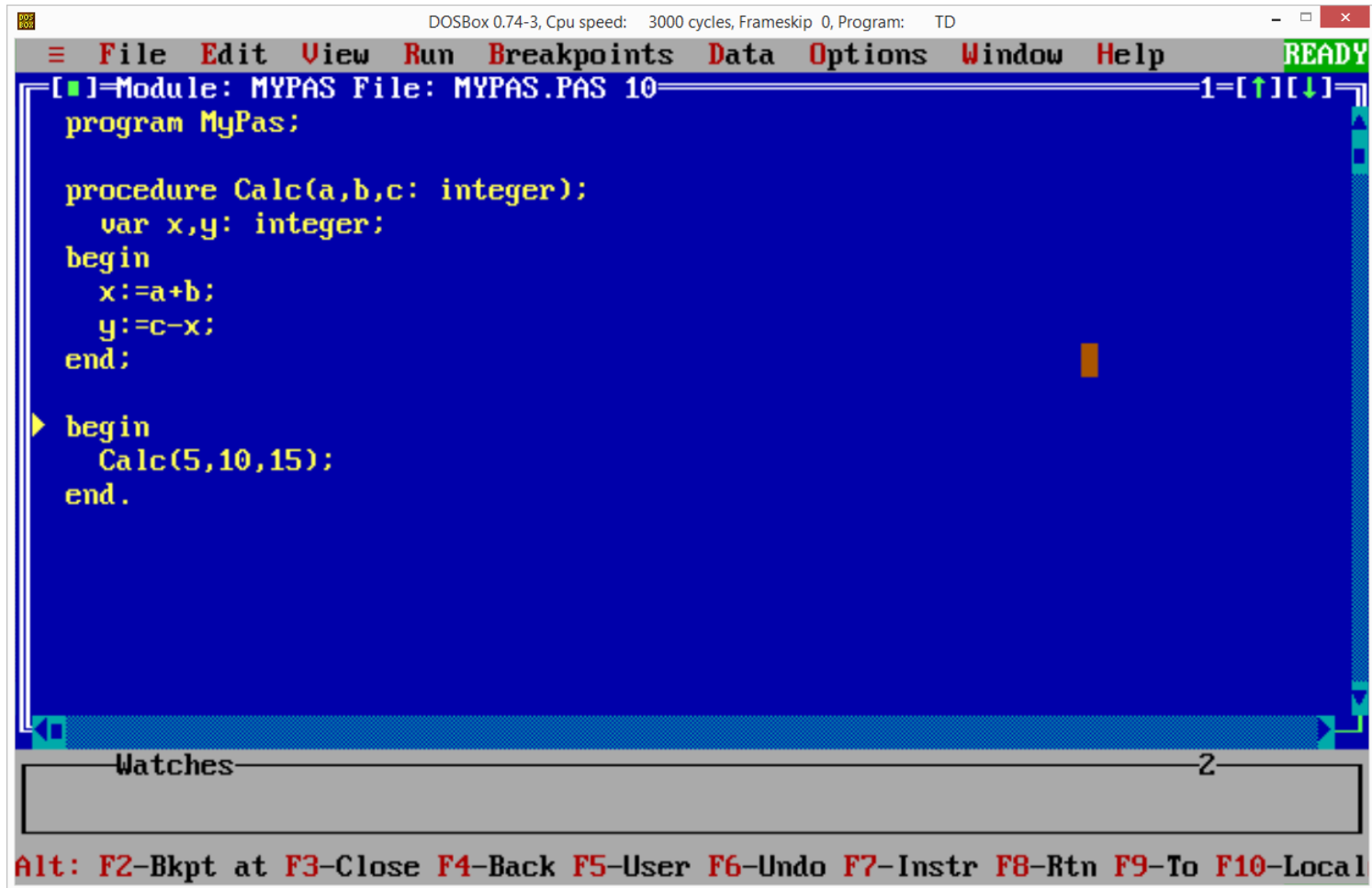
Опции отладчика

Options → Debugger



Загрузка в отладчик

td mypas.exe



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

[] Module: MYPAS File: MYPAS.PAS 10 1=[↑][↓]

```
program MyPas;  
  
  procedure Calc(a,b,c: integer);  
    var x,y: integer;  
  begin  
    x:=a+b;  
    y:=c-x;  
  end;  
  
  begin  
    Calc(5,10,15);  
  end.
```

Watches 2

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

[CPU 80486] 3 [READY]

```

MYPAS.10: begin
  cs:001E 9A0000374F    call    4F37:0000
  cs:0023 55            push    bp
  cs:0024 89E5          mov     bp,sp
MYPAS.11: Calc(5,10,15):
  cs:0026 B80500        mov     ax,0005
  cs:0029 50            push    ax
  cs:002A B80A00        mov     ax,000A
  cs:002D 50            push    ax
  cs:002E B80F00        mov     ax,000F
  cs:0031 50            push    ax
  cs:0032 E8CBFF        call    MYPAS.CALC
MYPAS.12: end.
  cs:0035 5D            pop     bp
  cs:0036 31C0          xor     ax,ax

```

ax	000F	c=0
bx	4F37	z=1
cx	0405	s=0
dx	4F37	o=0
si	0264	p=1
di	0150	a=0
bp	3FFE	i=1
sp	3FF8	d=0
ds	4F8B	
es	4F8B	
ss	4FB5	
cs	4F33	
ip	0032	

```

4F23:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
4F23:0008 AD DE E4 01 0C 1C AE 01 H 00000000
4F23:0010 0C 1C 80 02 67 16 D5 07 00000000
4F23:0018 01 01 01 00 02 FF FF FF 00000000
4F23:0020 FF FF FF FF FF FF FF FF

```

ss:4000	0000
ss:3FFE	0000
ss:3FFC	0005
ss:3FFA	000A
ss:3FF8	000F

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

[F5]=CPU 80486 ss:3FFC = 0005 3 [F5]

```

MYPAS.CALC: begin
  cs:0000 55          push    bp
  cs:0001 89E5        mov     bp,sp
  cs:0003 83EC04      sub     sp,0004
MYPAS.6:  x:=a+b;
  cs:0006 8B4608      mov     ax,[bp+08]
  cs:0009 034606      add     ax,[bp+06]
  cs:000C 8946FE      mov     [bp-02],ax
MYPAS.7:  y:=c-x;
  cs:000F 8B4604      mov     ax,[bp+04]
  cs:0012 2B46FE      sub     ax,[bp-02]
  cs:0015 8946FC      mov     [bp-04],ax
MYPAS.8:  end;
  cs:0018 89EC        mov     sp,bp
  cs:001A 5D          pop     bp
  
```

ax	000F	c=0
bx	4F37	z=0
cx	0405	s=0
dx	4F37	o=0
si	0264	p=1
di	0150	a=0
bp	3FF4	i=1
sp	3FF0	d=0
ds	4F8B	
es	4F8B	
ss	4FB5	
cs	4F33	
ip	0006	

```

4F23:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
4F23:0008 AD DE E4 01 0C 1C AE 01 H 00 00 00
4F23:0010 0C 1C 80 02 67 16 D5 07 00 00 00 = f
4F23:0018 01 01 01 00 02 FF FF FF 00 00 00
4F23:0020 FF FF FF FF FF FF FF FF
  
```

ss:3FF8	000F
ss:3FF6	0035
ss:3FF4	3FFE
ss:3FF2	7346
ss:3FF0	4F33

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local

Спасибо за внимание.