

# Жизненный цикл разработки низкоуровневой программы



# Текстовый файл исходной программы

---

- ❑ Файл с текстом исходной программы должен иметь расширение `.asm`
- ❑ Каждая строка текста должна заканчиваться `Enter`
- ❑ *Стандартный текстовый редактор.* Для правильного представления русскоязычных комментариев редактор должен поддерживать кодировку OEM (кодировка 866)
- ❑ При наборе строк программы на языке ассемблера **стремиться к большей читабельности, т.е.** структурировать строки

## Исходная программа. Файл metod.asm

; Получить 2-х байтную сумму двух однобайтных кодов в регистре AX

.386

---

```
dseg segment use16
```

```
    a    db 34
```

```
    b    db 75h
```

```
dseg ends
```

```
cseg segment use16
```

```
assume ds:dseg, cs:cseg
```

```
m1:mov cx, dseg
```

```
    mov ds, cx
```

; расширим байты до 2-х байтных в регистрах и сложим

```
    movzx ax, ds:a
```

```
    movzx bx, ds:b
```

```
    add  ax, bx
```

; завершение исполнения

```
    mov ah, 4ch
```

```
    int 21h
```

```
cseg ends
```

```
end m1
```

# Трансляция



- Задача транслятора - преобразовать символически записанные команды процессора и данные в машинные коды
- Машинные коды он записывает в файл с расширением **.obj** - «объектный файл»
- Листинг трансляции (файл с расширением **.lst**)- это текстовый протокол результатов трансляции, включая сообщения о синтаксических ошибках

## Протокол трансляции исходного текста - `metod.lst`

```
1 ; metod.asm - Получить          2-х байтную сумму двух однобайтных кодов
2                               .386
3   dseg segment use16
4   0000 22          a          db 34
5   0001 75          b          db 75h
6   0002          dseg ends
7   0000          cseg segment use16
8   0000          assume ds:dseg, cs:cseg
9   0000 B9 0000 s   m1:      mov cx, dseg
10  0003 8E D9          mov ds, cx
11                               ; расширим байты до 2-х байтных и сложим
12  0005 0F B6 06 0000 r   movzx ax, ds:a
13  000A 0F B6 1E 0001 r   movzx bx, ds:b
14  000F 03 C3          add ax, bx
15                               ; завершение исполнения
16  0011 B4 4C          mov ah, 4ch
17  0013 CD 21          int 21h
18  0015          cseg ends
19  end m1
```

*16-разрядные  
внутрисегментные адреса*

## Формат протокола трансляции

---

- Первая колонка - номера строк исходного текста.
- Вторая колонка - показывает *внутрисегментные адреса начала размещения команд и данных*. Для 16-разрядной программы внутрисегментный адрес - четыре hex – цифры.
- Третья колонка - *машинные коды команд и данных* в hex-виде.  
Символ S рядом с машинным кодом говорит лишь о том, что команда содержит указатель адреса сегмента памяти.  
Символ R после машинного кода команды говорит о том, что машинный код содержит внутрисегментный адрес операнда
- Четвертая колонка показывает транслируемую символическую строку из исходного текста программы

## Компоновка



- Компоновщик создает из машинного кода (.obj) «исполняемый код» для операционной системы. исполняемый код дополнен технической информацией о программных сегментах, необходимой ОС при загрузке кодов из программных сегментов в память
- Исполняемый код сохраняется в файле с расширением **.exe** или **.com**.

# Отладка

---

Для отладки исполняемого кода после загрузки его в память используются **программы-отладчики**.

Они позволяют:

- управлять процессором** во время исполнения команд программы:
  - исполнять команды с остановкой после каждой
  - исполнять команды до определенного адреса
  - возвращать процессор к уже исполненной команде и другое
- в моменты приостановки процессора **просматривать или изменять текущее содержимое байтов памяти и регистров процессора**
- и другое

# Инструментальные средства

---

Классический пакет Borland Turbo Assembler (TASM) включает в себя необходимые утилиты для трансляции, компоновки и отладки **16-разрядных программ** для процессоров x86:

- **Tasm.exe** – транслятор
- **Tlink.exe** – компоновщик
- **Td.exe** - отладчик

# Файловые менеджеры

---

- Программы TASM являются **консольными утилитами**, то есть требуют режима командной строки.
- Самой удобной средой для работы с этими утилитами является не чистый консольный режим, а среда **файловых менеджеров** (**FAR**, Total Commander, NC, VC и др.)
- В интерфейс файловых менеджеров интегрированы:
  - **консольная среда** командного режима,
  - **удобный панельный интерфейс** для работы с дисками, каталогами и файлами,
  - **встроенный текстовый редактор** (с кодовой страницей 866)

## Транслятор TASM

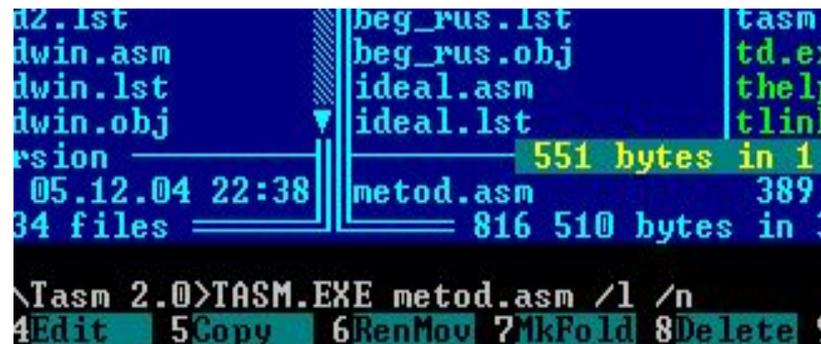
- Запуск транслятора Tasm.exe из командной строки:  
    > **tasm.exe metod.asm /l /n**

Объектный файл metod.obj по умолчанию будет создан в каталоге, где находится транслятор.

Ключи трансляции:

**/l** - создать файл-протокол трансляции

**/n** – не создавать в конце протокола список символических имен (для экономии бумаги при печати)



```
\Tasm 2.0>TASM.EXE metod.asm /l /n
4Edit 5Copy 6RenMov 7MkFold 8Delete 9

dir
 12.lst
 dwin.asm
 dwin.lst
 dwin.obj
ersion
05.12.04 22:38
34 files

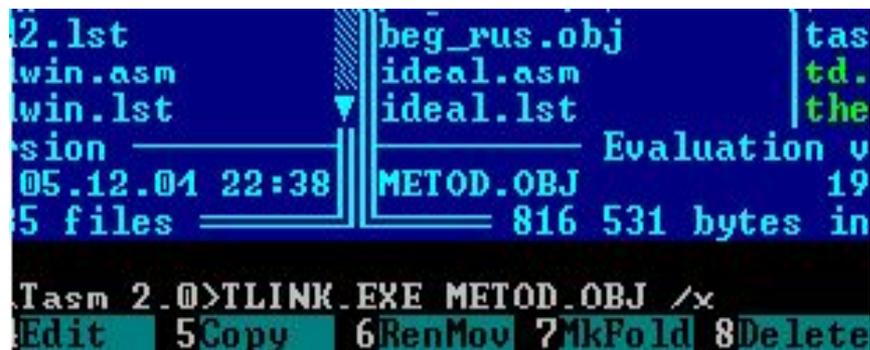
 beg_rus.lst
 beg_rus.obj
 ideal.asm
 ideal.lst
551 bytes in 1
metod.asm
389
816 510 bytes in 3
```

## Компоновщик TLINK

- Запуск компоновщика **Tlink.exe** из командной строки:  
    > **tlink.exe metod.obj /x**

Исполняемый файл `metod.exe` создается в каталоге, где был объектный модуль.

Ключ: `/x` - не создавать файл карты связи (`.map`)



```
2.lst
win.asm
win.lst
sion
05.12.01 22:38
5 files
beg_rus.obj
ideal.asm
ideal.lst
Evaluation v
METOD.OBJ
19
816 531 bytes in

Tasm 2.0>TLINK.EXE METOD.OBJ /x
Edit 5 Copy 6 RenMov 7 MkFold 8 Delete
```

## Отладчик Td.exe

- Загрузка отладчика из командной строки вместе с исполняемой программой:

› **td.exe metod.exe**

The screenshot shows the TD.EXE METOD.EXE debugger window. The main window displays assembly code with the following instructions:

```
cs:0000 B9A55F mov cx,5FA5
cs:0003 8ED9 mov ds,cx
cs:0005 0FB6060000 movzx ax,byte ptr [0000]
cs:000A 0FB61E0100 movzx bx,byte ptr [0001]
cs:000F 03C3 add ax,bx
cs:0011 B44C mov ah,4C
cs:0013 CD21 int 21
cs:0015 0000 add [bx+sil],al
cs:0017 0000 add [bx+sil],al
cs:0019 0000 add [bx+sil],al
cs:001B 0000 add [bx+sil],al
cs:001D 0000 add [bx+sil],al
cs:001F 0000 add [bx+sil],al
cs:0021 0000 add [bx+sil],al
cs:0023 0000 add [bx+sil],al
```

Registers window (Окно регистров) shows:

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	0000	d=0
ds	5F95	Флаги
es	5F95	
ss	5FA5	
cs	5FA6	
ip	0000	

Data segments window (Окно для сегментов данных) shows:

ds:0000	CD 20 FF 9F 00 9A F0 FE = Я ЪЕ	ss:0006	0000
ds:0008	1D F0 E0 01 60 26 00 01 +E0@i8v@	ss:0004	0000
ds:0010	6A 26 89 02	ss:0002	0000
ds:0018	01 01 01 00	ss:0000	7522

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