Весна 2021

# Системное программное обеспечение

Онлайн-лекции

## Лекция №9: Файлы листинга. Многомодульные программы

Доцент, к.т.н. ГОЛЬЦОВ Александр Геннадьевич

# Листинги

- В ходе ассемблирования может порождаться файл листинга с расширением LST
- В ходе компоновки может порождаться файл карты памяти с расширением MAP
- Для их интерпретации необходимо, в частности, разобраться с понятием группировки сегментов

# Директива GROUP

- Группировка сегментов логически соединяет сегменты в один, при этом смещения меток в сегментах группы исчисляются относительно начала группы, а сегментный регистр настраивается на группу
- (Имеются в виду сегменты - структурные единицы памяти программы, не сегментные регистры!)
- Формат:

**GROUP имя_группы сегмент1, сегмент2, ...**

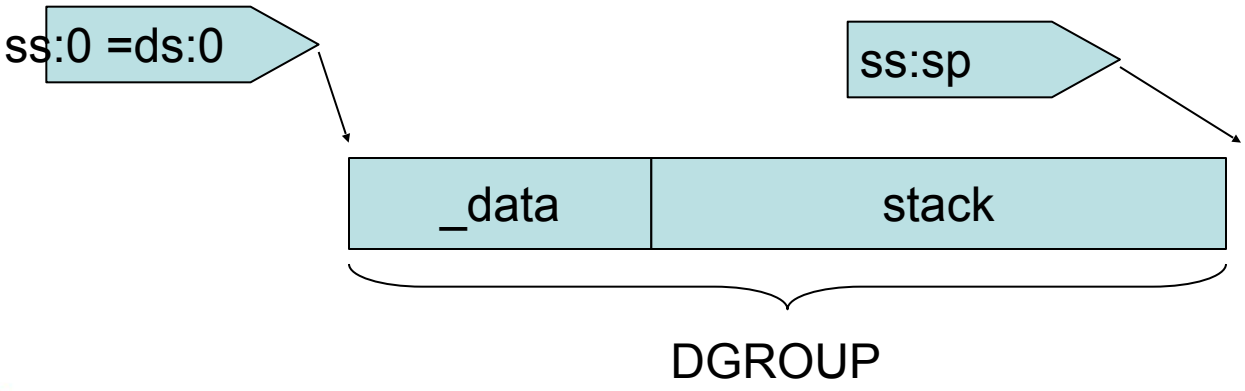
# Директива .STARTUP

- Генерирует код, настраивающий DS, SS и SP в соответствии с моделью
- Сегмент стека и данных по умолчанию объединены в группу DGROUP
- Директива .STARTUP перенастраивает SS и SP так, что
  SS = DS = начало группы DGROUP,
  SP = за конец группы DGROUP
  - это может сбивать с толку

# Пример .STARTUP

```
.model small
.stack 100h
.data
Hello db 'Hello, world$'
.code
start:
.startup

        mov     ax,4C00h
        int 21h
        end start
```

```
cs:0000 BA354F          mov     dx,4F35
cs:0003 8EDA            mov     ds,dx
cs:0005 8CD3            mov     bx,ss
cs:0007 2BDA            sub     bx,dx
cs:0009 D1E3            shl     bx,1
cs:000B D1E3            shl     bx,1
cs:000D D1E3            shl     bx,1
cs:000F D1E3            shl     bx,1
cs:0011 FA              cli
cs:0012 8ED2            mov     ss,dx
cs:0014 03E3            add     sp,bx
cs:0016 FB              sti
```



# "Побочные продукты" компиляции

- Имеем исходный текст my.asm
- Ассемблирование:
  tasm my.asm ➔ my.obj (или ошибки)
  /L ➔ my.lst
- Сборка:
  tlink my.obj ➔ my.exe (или ошибки)
  ➔ my.map
- Файлы листинга и карты сегментов содержат дополнительную полезную информацию о процессе ассемблирования и сборки программы
- Там, в частности, есть сообщения об ошибках

# Опции ассемблирования, влияющие на листинг

- Листинг порождает TASM
- /c - включать в листинг информацию о перекрестных ссылках (какие символы в каких строках определены и упомянуты)
- /L - генерировать обычный листинг
- /LA - генерировать расширенный листинг
- /n - не включать в листинг таблицы символов
- /x - принудительно включать в листинг то, что исключается условиями условной компиляции

# Задание опций компиляции

- Прямо **в командной строке** TASM - если опций много, набирать каждый раз неудобно
- Запуск можно вынести **в пакетный файл** T.BAT:
  ```
  tasm /l/c/m2 %1
  ```
  вызов: T my.asm
- **В файле TASM.CFG** - перечисляются ключи со слэшами, концы строк трактуются как пробелы
- **В текстовом файле** (MYOPTN.TXT), при этом ссылка на файл дается в командной строке:
  ```
  tasm @myoptn.txt my.asm
  ```
- Аналогично - для TLINK (в т.ч. **tlink.cfg**)

# Исходный текст

```
.model small
.stack 100h
.data
Hello db 'Hello, world$'
.code
start:  mov     ax,@data
        mov     ds,ax

        mov     dx, offset Hello

        mov     ax,4C00h
        int 21h
        end start
```

tasm /l my

# Основной листинг

```
Turbo Assembler  Version 3.2        04/12/21 20:12:56       Page 1
my.ASM


      1 0000                        .model small
      2 0000                        .stack 100h
      3 0000                        .data
      4 0000  48 65 6C 6C 6F 2C 20+ Hello db 'Hello, world$'
      5       77 6F 72 6C 64 24
      6 000D                        .code
      7 0000  B8 0000s              start:  mov     ax,@data
      8 0003  8E D8                         mov     ds,ax
      9
     10 0005  BA 0000r                      mov     dx, offset Hello
     11
     12 0008  B8 4C00                       mov     ax,4C00h
     13 000B  CD 21                         int 21h
     14                                     end start
```

NN строк | смещ. | машинный код | исходный текст

s (segment) = "сегмент определяется загрузчиком"
r (relocatable) = "смещение может измениться при компоновке"

# Таблица символов

```
Symbol Name          Type    Value

??DATE               Text    "04/12/21"
??FILENAME           Text    "my       "
??TIME               Text    "20:12:56"
??VERSION            Number  0314
@32BIT               Text    0
@CODE                Text    _TEXT
@CODESIZE            Text    0
@CPU                 Text    0101H
@CURSEG              Text    _TEXT
@DATA                Text    DGROUP
@DATASIZE            Text    0
@FILENAME            Text    MY
@INTERFACE           Text    00H
@MODEL               Text    2
@STACK               Text    DGROUP
@WORDSIZE            Text    2
HELLO                Byte    DGROUP:0000
START                Near    _TEXT:0000
```

# Таблица сегментов

```
Groups & Segments  Bit  Size  Align   Combine  Class

DGROUP             Group
  STACK            16   0100  Para    Stack    STACK
  _DATA            16   000D  Word    Public   DATA
_TEXT              16   000D  Word    Public   CODE
```

**Комбинирование** - как комбинировать одноименные сегменты разных модулей (файлов .asm/.obj многомодульной программы):
- private - не объединяются,
- public - объединяются,
- common - перекрываются, начинаясь с одного адреса (размер равен размеру самого большого)
- stack - как public, дополнительно загрузчик настраивает на такой сегмент ss:sp (иначе - инициализировать вручную как ds)

# Перекрестные ссылки



```
Symbol Name                Type    Value                          Cref (defined at #)

??DATE                     Text    "04/12/21"
??FILENAME                 Text    "my      "
??TIME                     Text    "20:43:20"
??VERSION                  Number  0314
@32BIT                     Text    0                              #1
@CODE                      Text    _TEXT                          #1  #1  #6
@CODESIZE                  Text    0                              #1
@CPU                       Text    0101H
@CURSEG                    Text    _TEXT                          #3  #6
@DATA                      Text    DGROUP                         #1  7
@DATASIZE                  Text    0                              #1
@FILENAME                  Text    MY
@INTERFACE                 Text    00H                            #1
@MODEL                     Text    2                              #1
@STACK                     Text    DGROUP                         #1
@WORDSIZE                  Text    2                              #3  #6
HELLO                      Byte    DGROUP:0000                    #4  10
START                      Near     _TEXT:0000                    #7  14

Groups & Segments          Bit Size Align  Combine Class          Cref (defined at #)

DGROUP                     Group                                  #1  1  7
  STACK                    16  0100 Para   Stack   STACK          #2
  _DATA                    16  000D Word   Public  DATA           #1  #3
_TEXT                      16  000D Word   Public  CODE           #1  1  #6  6
```

# Листинг и макросы

- При наличии в программе макросов процесс их "разворачивания" в коде отображается в листинге
- Это происходит каждый раз при использовании макроса
- Если макрос сложный (как "умное умножение") из прошлой лекции, а листинг расширенный (опция /LA), то подробно раскрываются изменения в управляющих переменных, и листинг вызова макроса занимает несколько страниц

# Файл карты сегментов

- Расширение .MAP
- Генерируется компоновщиком (TLINK), если не запрещен
- Опции:
  **/m** - включать символы PUBLIC (для многомодульных программ)
  **/s** - подробная карта сегментов
  **/x** - запретить генерировать карту
- Информация весьма актуальна для многомодульных программ
- Содержит адрес точки входа в программу

# Простой файл карты сегментов (tlink /m /s my.obj)

```
 Start  Stop   Length Name               Class

 00000H 0000CH 0000DH _TEXT             CODE
 00010H 0001CH 0000DH _DATA             DATA
 00020H 0011FH 00100H STACK             STACK


Detailed map of segments

 0000:0000 000D C=CODE   S=_TEXT         G=(none)  M=MY.ASM     ACBP=48
 0001:0000 000D C=DATA   S=_DATA         G=DGROUP  M=MY.ASM     ACBP=48
 0001:0010 0100 C=STACK  S=STACK         G=DGROUP  M=MY.ASM     ACBP=74

 Address         Publics by Name


 Address         Publics by Value

Program entry point at 0000:0000
```

Используются условные адреса сегментов, но предположительный порядок сегментов в памяти будет таким

# Некоторые пояснения

- Start и Stop в обычной карте сегментов - смещения в исполнимом файле после заголовка, с которых начинаются коды сегментов.
- Сегменты данных и стека по умолчанию сгруппированы в DGROUP, но если мы не писали .STARTUP, то сегментные регистры DS и SS настроены на каждый сегмент в отдельности.
- Группировка стека и сегмента данных может рассматриваться как формальный момент

# ACBP

Сокращение ACBP расшифровывается как A (выравнивание) и C (комбинирование). Эта информация занимает четыре бита. TLINK использует только два из них - A и C. В файле .MAP эта информация приводится в шестнадцатеричном виде. Значения полей в ACBP - приведены ниже:

| Поле | Значение | Описание |
|---|---|---|
| Поле A (Выравнивание) | 00 | Абсолютный сегмент |
| | 20 | Сегмент, выровненный на границу байта |
| | 40 | Сегмент, выровненный на границу слова |
| | 60 | Сегмент, выровненный на границу параграфа |
| | 80 | Сегмент, выровненный на границу страницы |
| | A0 | неименованный абсолютный участок памяти |
| Поле C (Комбинирование) | 00 | Не комбинируется |
| | 08 | Общий комбинируемый сегмент |

# Многомодульная программа

- Множество исходных текстов, множество файлов .OBJ
- Объектные файлы могут распространяться авторами без исходных текстов
- Процесс компиляции:

  tasm src1.asm ➔ src1.obj
  tasm src2.asm ➔ src2.obj
  tlink src1 src2 obj3 ➔ src1.exe
- Можно так (разделитель - точка с запятой):

  tasm src1; src2 ➔ src1.obj src2.obj
- Можно так (запятая, имя исполнимого файла):

  tlink src1 src2, result ➔ result.exe

# Замечание

Не называйте файлы многомодульного проекта просто числами, например, 1.asm и 2.asm.

Компоновщик этого не поймет.

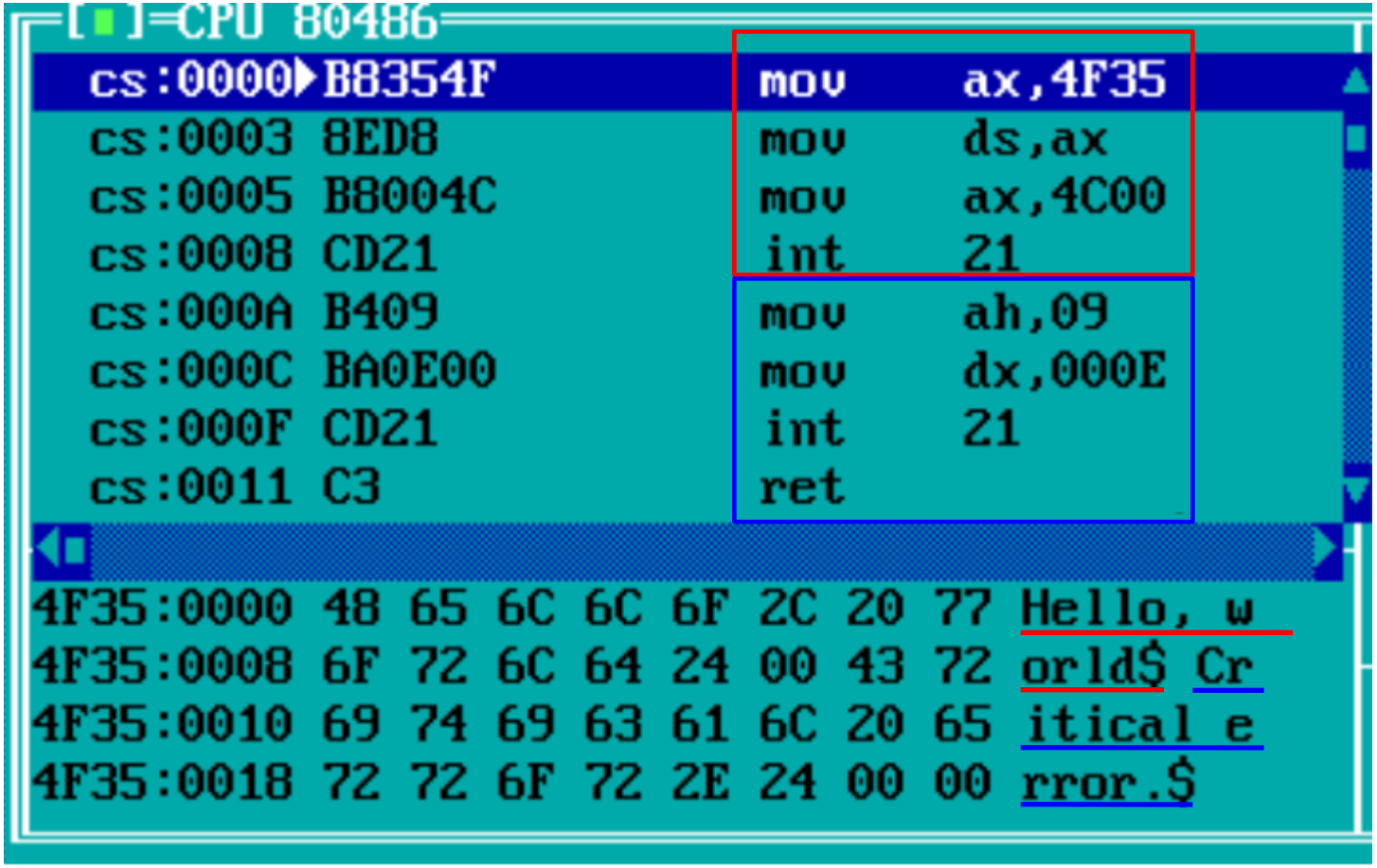
Используйте имена файлов, начинающиеся с буквы или разрешенного символа - не цифры.

# Два модуля asm1 и asm2

```
.model small
.stack 100h



.data
Hello db 'Hello, world$'

.code
start:  mov     ax,@data
        mov     ds,ax



        mov     ax,4C00h
        int 21h
        end start
```

```
.model small
.stack 100h
.data
s db 'Critical error.$'

.code
ErrMsg proc
        mov  ah,9
        mov  dx,offset Hello
        int 21h
        ret
ErrMsg endp


        end
```

# И что в результате?



```
Turbo Assembler  Version 3.2        04/12/21 22:13:37      Page 1
asm1.ASM



      1 0000                        .model small
      2 0000                        .stack 100h
      3 0000                        .data
      4 0000  48 65 6C 6C 6F 2C 20+ Hello db 'Hello, world$'
      5       77 6F 72 6C 64 24
      6 000D                        .code
      7 0000  B8 0000s              start:  mov     ax,@data
      8 0003  8E D8                         mov     ds,ax
      9
     10 0005  B8 4C00                       mov     ax,4C00h
     11 0008  CD 21                         int 21h
     12                                     end start
♀Turbo Assembler  Version 3.2        04/12/21 22:13:37      Page 2
Symbol Table

Symbol Name                           Type   Value

HELLO                                 Byte   DGROUP:0000
START                                 Near   _TEXT:0000

Groups & Segments                     Bit Size Align  Combine Class

DGROUP                                Group
  STACK                               16  0100 Para   Stack   STACK
  _DATA                               16  000D Word   Public  DATA
_TEXT                                 16  000A Word   Public  CODE
```

```
Turbo Assembler  Version 3.2        04/12/21 22:13:42       Page 1
asm2.ASM

      1 0000                        .model small
      2 0000                        .stack 100h
      3 0000                        .data
      4 0000  43 72 69 74 69 63 61+ s db 'Critical error.$'
      5       6C 20 65 72 72 6F 72+
      6       2E 24
      7 0010                        .code
      8 0000                        ErrMsg proc
      9 0000  B4 09                         mov  ah,9
     10 0002  BA 0000r                      mov  dx,offset s
     11 0005  CD 21                         int 21h
     12 0007  C3                            ret
     13 0008                        ErrMsg endp
     14                                     end
♀Turbo Assembler Version 3.2        04/12/21 22:13:42       Page 2
Symbol Table

Symbol Name                     Type   Value

ERRMSG                          Near   _TEXT:0000
S                               Byte   DGROUP:0000

Groups & Segments               Bit Size Align  Combine Class

DGROUP                          Group
  STACK                         16  0100 Para   Stack   STACK
  _DATA                         16  0010 Word   Public  DATA
_TEXT                           16  0008 Word   Public  CODE
```

```
 Start  Stop   Length Name               Class

 00000H 00011H 00012H _TEXT             CODE
 00020H 0003DH 0001EH _DATA             DATA
 00040H 0023FH 00200H STACK             STACK


Detailed map of segments

 0000:0000 000A C=CODE   S=_TEXT          G=(none)  M=ASM1.ASM   ACBP=48
 0000:000A 0008 C=CODE   S=_TEXT          G=(none)  M=ASM2.ASM   ACBP=48
 0002:0000 000D C=DATA   S=_DATA          G=DGROUP  M=ASM1.ASM   ACBP=48
 0002:000E 0010 C=DATA   S=_DATA          G=DGROUP  M=ASM2.ASM   ACBP=48
 0002:0020 0100 C=STACK  S=STACK          G=DGROUP  M=ASM1.ASM   ACBP=74
 0002:0120 0100 C=STACK  S=STACK          G=DGROUP  M=ASM2.ASM   ACBP=74


Program entry point at 0000:0000
```

# А как взаимодействовать?

- Коды сегментов склеились, код процедуры ErrMsg есть в исполнимом файле.
- Чтобы им воспользоваться в модуле Asm1, в модуле Asm2 его нужно опубликовать - сделать доступным во всей программе.
- А в других модулях необходимо описать этот символ как внешний.
- Разбираться, есть ли внешние символы среди опубликованных будет компоновщик (TLINK), TASM этим не занимается
- Наличие других модулей TASMу не требуется

# Директива EXTRN

- Декларирует имя (метку, имя процедуры, константу) как описанную во внешнем модуле и позволяет генерировать объектный файл, ссылающийся на такое имя и требующий разрешения (resolve) ссылки.
- Синтаксис:

  extrn имя:тип, имя:тип, .....
- В качестве типа указываются:
  - byte, word, dword... - для меток данных
  - near, far - для меток кода
  - abs - для констант, объявленных через EQU

# Директива PUBLIC

- Перечисляет метки и символьные имена, которые необходимо сделать доступными в других модулях
- Синтаксис (в простейшем случае):
  PUBLIC имя, имя, ....

# Два модуля связаны

```
.model small
.stack 100h

EXTRN s: byte, ErrMsg: near

.data
Hello db 'Hello, world$'
        PUBLIC Hello
.code
start:  mov     ax,@data
        mov     ds,ax

        call    ErrMsg

        mov     ax,4C00h
        int 21h
        end start
```

```
.model small
.stack 100h
.data
s db 'Critical error.$'
        EXTRN Hello: byte
.code
ErrMsg proc
        mov  ah,9
        mov  dx,offset Hello
        int 21h
        ret
ErrMsg endp

PUBLIC s, ErrMsg
        end
```

```
 Start  Stop   Length Name               Class

 00000H 00015H 00016H _TEXT             CODE
 00020H 0003DH 0001EH _DATA             DATA
 00040H 0023FH 00200H STACK             STACK


Detailed map of segments

 0000:0000 000D C=CODE   S=_TEXT         G=(none)  M=ASM1.ASM   ACBP=48
 0000:000E 0008 C=CODE   S=_TEXT         G=(none)  M=ASM2.ASM   ACBP=48
 0002:0000 000D C=DATA   S=_DATA         G=DGROUP  M=ASM1.ASM   ACBP=48
 0002:000E 0010 C=DATA   S=_DATA         G=DGROUP  M=ASM2.ASM   ACBP=48
 0002:0020 0100 C=STACK  S=STACK         G=DGROUP  M=ASM1.ASM   ACBP=74
 0002:0120 0100 C=STACK  S=STACK         G=DGROUP  M=ASM2.ASM   ACBP=74

  Address         Publics by Name

 0000:000E       ERRMSG
 0002:0000       HELLO
 0002:000E       S

  Address         Publics by Value

 0000:000E       ERRMSG
 0002:0000       HELLO
 0002:000E       S

Program entry point at 0000:0000
```

# Где располагать EXTRN и PUBLIC?

- Директивы могут быть в любом месте модуля.
- Они всего лишь инструктируют ассемблер "поставить галочку" про доступность символьного имени.
- Реальное расположение метки (в каком сегменте, по какому смещению) определяется ее объявлением в том модуле, где она объявлена.
- EXTRN после использования символа или PUBLIC до его объявления могут потребовать компиляции в два прохода

# Другая модель памяти

- Если поменять SMALL на LARGE - программа станет некорректной, хотя компилируется и собирается успешно.
- В модели LARGE кодовые сегменты не объединяются, в программе два сегмента кодов, ближний вызов некорректен

# Для ErrMsg:near → far

```
 Start  Stop   Length Name                   Class

00000H 0000EH 0000FH ASM1_TEXT             CODE
00010H 00017H 00008H ASM2_TEXT             CODE
00020H 0003DH 0001EH _DATA                 DATA
00040H 0023FH 00200H STACK                 STACK


Detailed map of segments

0000:0000 000F C=CODE    S=ASM1_TEXT      G=(none)   M=ASM1.ASM    ACBP=48
0001:0000 0008 C=CODE    S=ASM2_TEXT      G=(none)   M=ASM2.ASM    ACBP=48
0002:0000 000D C=DATA    S=_DATA          G=DGROUP   M=ASM1.ASM    ACBP=48
0002:000E 0010 C=DATA    S=_DATA          G=DGROUP   M=ASM2.ASM    ACBP=48
0002:0020 0100 C=STACK   S=STACK          G=DGROUP   M=ASM1.ASM    ACBP=74
0002:0120 0100 C=STACK   S=STACK          G=DGROUP   M=ASM2.ASM    ACBP=74

  Address         Publics by Name

0001:0000       ERRMSG
0002:0000       HELLO
0002:000E       S
```

# Связь с программами на ЯВУ

- Объектные файлы, полученные путем ассемблирования, могут быть использованы при компоновке программ на ЯВУ.
- Объектные файлы, полученные при компиляции модулей на ЯВУ, могут быть собраны компоновщиком TLINK вместе с ассемблерными.
- Есть дополнительные ограничения, что видимо и что возможно экспортировать-импортировать при совместном использовании ассемблера и ЯВУ.
- Есть ограничения, какие регистры разрешено портить, при написании ассемблерных процедур для ЯВУ.

# Пример связи Паскаль - TASM

```
.model medium
.code
OutStr  proc far
        arg  sptr: dword
        push bp
        mov  bp, sp
        push ds

        mov  ah,9
        lds  dx,sptr
        int  21h

        pop ds
        pop bp
        ret 4
OutStr  endp

PUBLIC OutStr
        end
```

```
{$F+}
program MyPas;

procedure OutStr(sp:pointer); external;
{$L pas.obj}

var s:string;

begin
  s:='Hello$';
  OutStr(@s[1]);
end.
```

1. TASM pas.asm ➔ pas.obj

2. Затем компилируем программу на Паскале.

{$F+} означает Force far calls
{$L ...} могут быть где угодно, удобно писать их рядом с **external**.

# Спасибо за внимание.