

Analysis of debugging process. **Segmentation directives.**

Listing of Example Program Translation

Attributes of directive **segment**

How is a program compiled in TASM?

The program translation is executed by TASM program *tasm.exe*. this program is loaded with help of command string

TASM[options] name of initial file [,name of object file] [,name of listing] [,name of file of cross-references]

or more simply

TASM[options] name of initial file, , ,

(gray-colored elements are not compulsory).

Example:

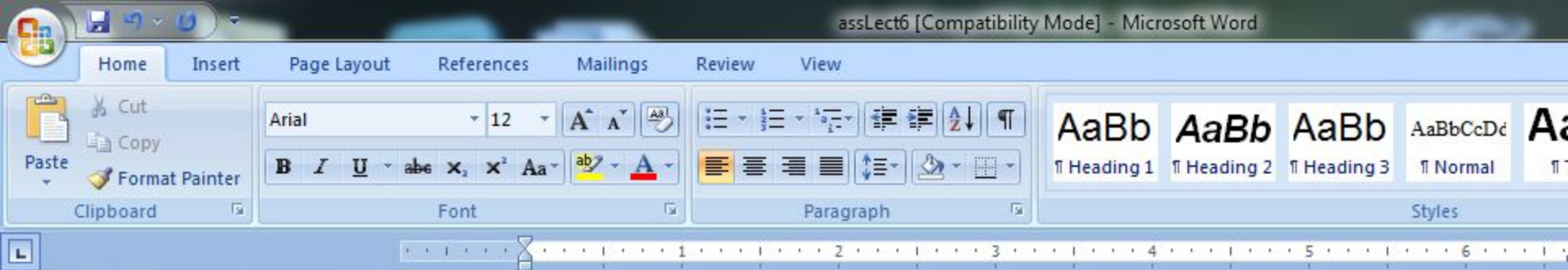
TASM /z/zi/n PRG0.asm,prg0.obj,prg0.lst,prg0.crf

or

TASM /z/zi/n PRG0.ASM, , ,

What is meant by listing file in assembly?

A **listing file** shows precisely how the **assembler** translates your source **file** into machine code. It's a report of translation.



Turbo Assembler Version 4.1 14/10/03 13:04:09 Page 1
 p0.ASM

Numbers of Assembler sentences

Offsets of Instructions relatively the Segment beginning

Machinery codes of instructions

The initial text of the program

↓
 ↓
 ↓

```

1
2 0000
3 0000 B8 0000s
4 0003 8E D8
5 0005 B4 09
6 0007 BA 0000r
7 000A CD 21
8 000C B8 4C00
9 000F CD 21
10 0011
11 0000
12 0000 54 68 65 20 70 72 6F+ MSG db "The program works !$"
13 67 61 6D 20 77 6F 72+
14 6B 73 21 24
15 0012
16 0000
```

```

assume CS:CODE, DS:DATA
CODE segment
    BEGIN: mov AX,DATA
           mov DS,AX
           mov AH,    09h
           mov DX,    offset MSG
           int 21h
           mov AX,    4C00h
           int 21h
CODE ends
DATA segment
DATA ends
STK segment stack
```

What are the main components (columns) of the program listing?

Program's listing includes four components (columns):

- 1. Numbers of Assembler sentences;**
- 2. Offsets of Instructions relatively the Segment beginning;**
- 3. Machinery codes of instructions;**
- 4. The initial text of the program.**

Syntax

name **SEGMENT** **[[align]]** **[[combine]]** **[[use]]** **[[‘class’]]**

Statements

Statements

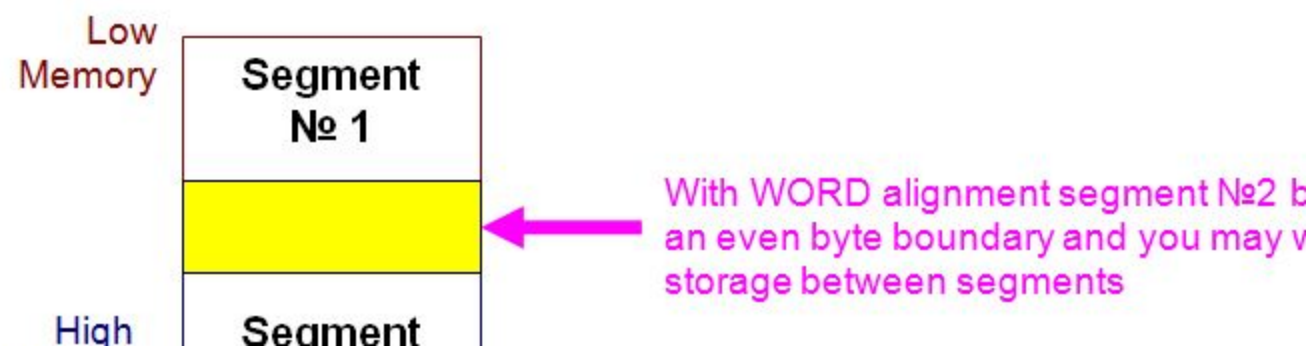
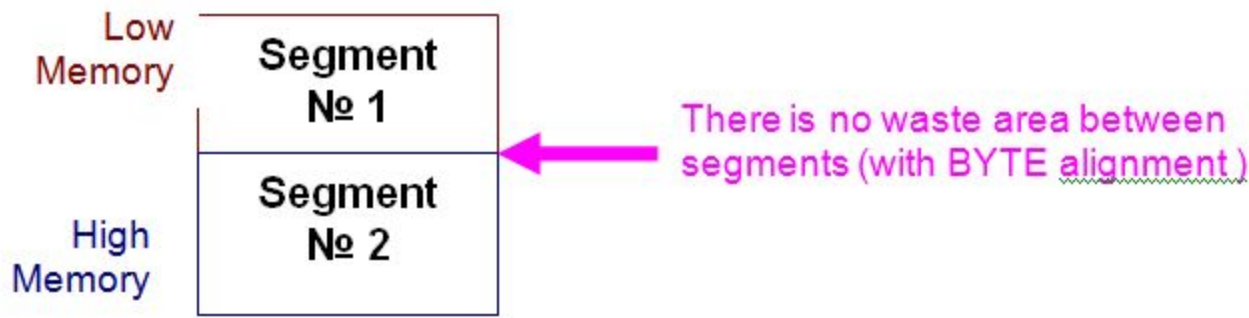
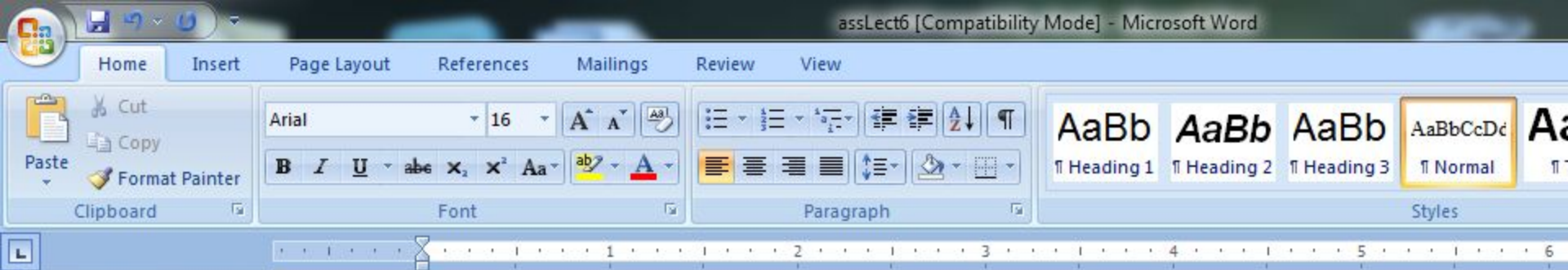
.....

Statements

Statements

name **ENDS**

- **Attribute of segment alignment (type of alignment).** It informs the compiler, that it is necessary to provide an allocation of the segment beginning at the given border (it is important for simplification of access to the data). There are following permissible meanings of this attribute:
 - **BYTE.** In this case the segment may be allocated starting from an arbitrary address in the main memory;
 - **WORD.** In this situation the initial (base) address of the segment will be dividable by 2;
 - **DWORD.** The segment begins with an address dividable by 4;
 - **PARA.** The segment begins with an address dividable by 16;
 - **PAGE.** The segment begins with an address dividable by 256;
 - **MEMPAGE.** The segment begins with an address dividable by 4 Kbytes.



Attribute of segments combining

(combining type). It informs the compiler, that it is necessary to combine segments of different modules, which have the same name. The default the combining is **PRIVATE**. Meanings of this attribute may be following:

PRIVATE. In this case the segment will not be united with other segments;

PUBLIC. It compels the compiler to unite all segments (with the same name) in one continues segment;

COMMON. In this case all the segments will overlay and use collectively (share) the memory (the size of such segment will be equal to the size of the segment with the largest area).

STACK. It determines a segment for the stack (there is some analogy with the **PUBLIC**).

Attribute of segment class (type of class). It is enclosed in inverted commas (") string, which helps the compiler **to determine an order of segments location** during compilation a program, which consists from several modules

Attribute of segment's size. For

microprocessors i80386 and higher segments may be 16- or 32-width, so it may influence on segments sizes and on the order of the physical address formation inside these segments. This attribute may have the following meanings:

USE16 – it means, that during a physical address formation, can be used only 16-width offset;

USE32 – it means, that during a physical address formation, can be used only 32-width offset.