Chapter 1

Real Mode Memory Models

The opposite of the medium model is the compact model. Here the total function code cannot exceed 64KiB, but there is more space for global data, stack and heap. The large model can deal with function code and data larger than 64KiB, but is slow in execution as only far pointers are used for both code and data. Borland C++ limits the size of all global data to 64KiB. The huge memory model sets aside that limit, allowing global data to occupy more than 64KiB[1].

---

[1]Also here the same limitation as before: If the source file is too big to fit into one 64KiB data segment, the programmer must break it up into different source files and compile each file separately.
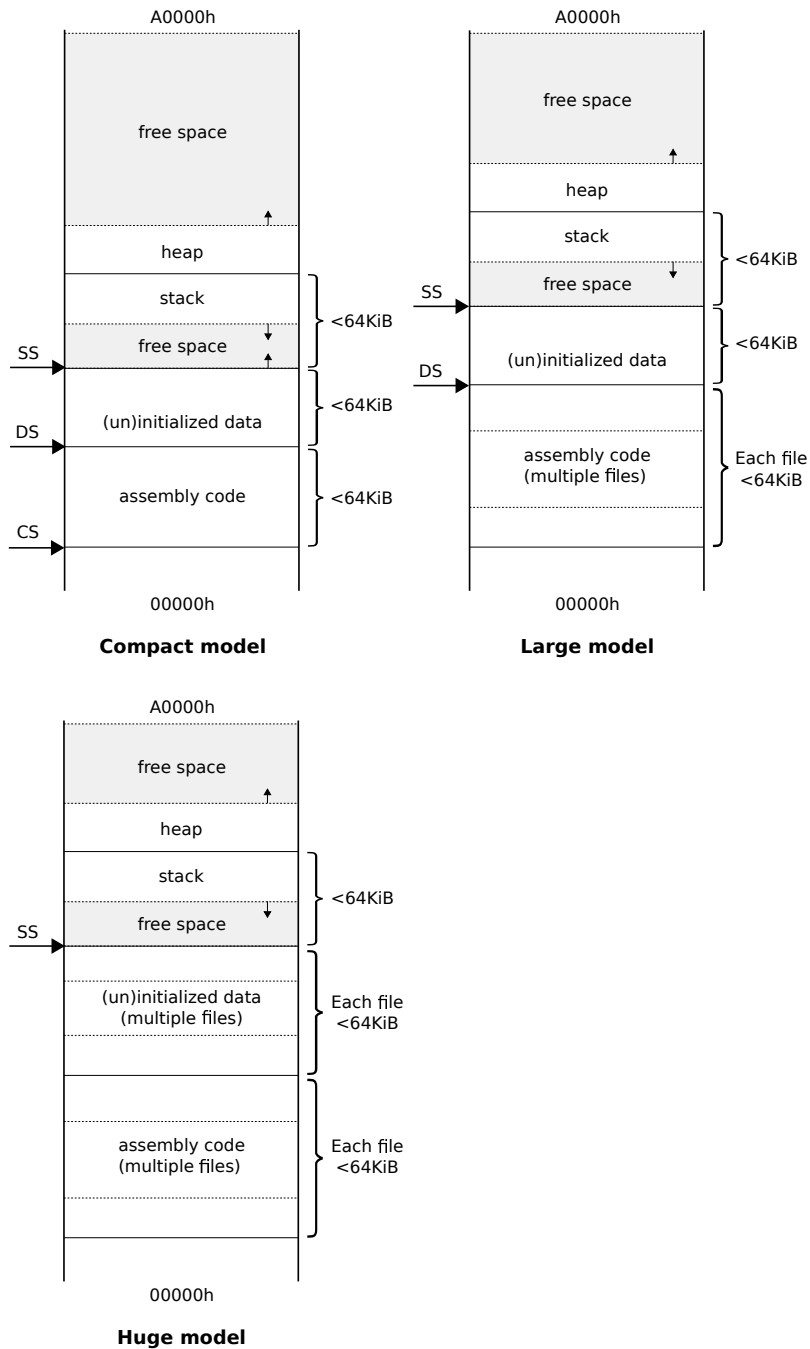
A0000h

free space

heap

stack

free space   <64KiB

SS

(un)initialized data   <64KiB

DS

assembly code   <64KiB

CS

00000h

Compact model

A0000h

free space

heap

stack

free space   <64KiB

SS

(un)initialized data   <64KiB

DS

assembly code
(multiple files)   Each file
<64KiB

00000h

Large model

A0000h

free space

heap

stack

free space   <64KiB

SS

(un)initialized data
(multiple files)   Each file
<64KiB

assembly code
(multiple files)   Each file
<64KiB

00000h

Huge model

Figure 1.1: Compact, Large and Huge memory model.

Trivia : Although the name implies differently, the huge memory model in Borland C++ is still limited to segments up to 64KiB using far pointers, and not using huge pointers. The Watcom compiler, which gained popularity in the 90's, was able to break the 64KiB segment barrier using the huge pointer reference for the huge memory model[2].

---

[2]See https://open-watcom.github.io/open-watcom-v2-wikidocs/clr.html