.bss

In computer programming, the name **.bss** or **bss** is used by many compilers and linkers for a part of the data segment containing statically-allocated variables represented solely by zero-valued bits initially (i.e., when execution begins). It is often referred to as the "bss section" or "bss segment".

In C, statically-allocated variables without an explicit initializer are initialized to zero (for arithmetic types) or a null pointer (for pointer types). Implementations of C typically represent zero values and null pointer values using a bit pattern consisting solely of zero-valued bits (though this is not required by the C standard). Hence, the bss section typically includes all uninitialized variables declared at the file level (i.e., outside of any function) as well as uninitialized local variables declared with the `static` keyword. An implementation may also assign statically-allocated variables initialized with a value consisting solely of zero-valued bits to the bss section.

Typically, the program loader initializes the memory allocated for the bss section when it loads the program. Operating systems may use a technique called zero-fill-on-demand to efficiently implement the bss segment (McKusick & Karels 1986). In embedded software, the bss segment is mapped into memory that is initialized to zero by the C run-time system before `main()` is entered.

Some application binary interfaces also support an **sbss** segment for "small data". Typically, these data items can be accessed by leaner code using instructions that can only access a certain range of addresses.

Historically, **BSS** (from **Block Started by Symbol**) was a pseudo-operation in UA-SAP (United Aircraft Symbolic Assembly Program), the assembler developed in the mid-1950s for the IBM 704 by Roy Nutt, Walter Ramshaw, and others at United Aircraft Corporation.[1][2] The BSS keyword was later incorporated into FAP (FORTRAN Assembly Program), IBM's standard assembler for its 709 and 7090/94 computers. It defined a label (i.e. symbol) and reserved a block of uninitialized space for a given number of words (Timar 1996).

Peter van der Linden, a C programmer and author, says, "Some people like to remember it as 'Better Save Space.' Since the BSS segment only holds variables that don't have any value yet, it doesn't actually need to store the image of these variables. The size that BSS will require at runtime is recorded in the object file, but BSS (unlike the data segment) doesn't take up any actual space in the object file."[3]

References

- Stevens, W. Richard (1992). *Advanced Programming in the Unix Environment*. Addison–Wesley. Section 7.6. ISBN 0-201-56317-7.
- Timar, Ted; et al. (1996). "Unix - Frequently Asked Questions (1/7)" [4]. Question 1.3.
- McKusick, Marshall Kirk; Karels, Michael J. (1986). "A New Virtual Memory Implementation for Berkeley UNIX" [5]. University of California, Berkeley. p. 3.

References

[1] *Network Dictionary*. Javvin Press, 2007, p. 70.

[2] Coding for the MIT-IBM 704 Computer October 1957, p. V-10 (http://bitsavers.org/pdf/mit/computer_center/Coding_for_the_MIT-IBM_704_Computer_Oct57.pdf)

[3] Peter van der Linden, Expert C Programming: Deep C Secrets (http://books.google.com/books?id=4vm2xK3yn34C&pg=PA141), Prentice Hall 1994, p. 141

[4] http://www.faqs.org/faqs/unix-faq/faq/part1/section-3.html

[5] http://docs.freebsd.org/44doc/papers/newvm.pdf

Article Sources and Contributors

.bss *Source*: http://en.wikipedia.org/w/index.php?oldid=544926636 *Contributors*: Abdull, Alt-sysrq, Ameer.mawia, Andreas Kaufmann, AndrewHowse, Benhoyt, Breezeight, Chowbok, Chris Chittleborough, DantheCowMan, DmitTrix, Furrykef, Giraffedata, GottaFixThis, Greenstruck, Infovarius, Kimdime, Michael Drüing, Misternuvistor, Moocha, Neelix, SAE1962, Savago, Vadmium, Wernher, 39 anonymous edits

License