WIKIPEDIA

31-bit

In computer architecture, **31-bit** integers, memory addresses, or other data units are those that are 31 bits wide.

In 1983, IBM introduced 31-bit addressing in the System/370-XA mainframe architecture as an upgrade to the 24-bit physical and virtual,[1] and transitional 24-bit-virtual/26-bit physical,[2][3] addressing of earlier models.[4][5] This enhancement allowed address spaces to be 128 times larger, permitting programs to address memory above 16 MB (referred to as "above the line").[6][1] Support for COBOL, FORTRAN and later on Linux/390 were included.

Contents

Architecture

In the System/360, other than the 360/67, and early System/370 architectures, the general purpose registers were 32 bits wide, the machine did 32-bit arithmetic operations, and addresses were always stored in 32-bit words, so the architecture was considered 32-bit, but the machines ignored the top 8 bits of the address resulting in 24-bit addressing. With the XA extension, only the high order bit (bit 0) in the word was ignored for addressing. An exception is that mode-switching instructions also used bit 0. There were at least two reasons that IBM did not implement the 32-bit addressing of the 360/67

1. The loop control instructions **BXH** and **BXLE** did signed comparisons.
2. Much of the existing software used bit 0 as an end-of-list indicator.[7]

Transition

The transition was tricky: assembly language programmers, including IBM's own operating systems architects and developers, had been using the spare byte at the top of addresses for flags for almost twenty years.[8] IBM chose to provide two forms of addressing to minimize the pain: if the most significant bit (bit 0) of a 32-bit address was on, the next 31 bits were interpreted as the virtual address. If the most significant bit was off, then only the lower 24 bits were treated as the virtual address (just as with pre-XA systems). Thus programs could continue using the seven low-order bits of the top byte for other purposes as long as they left the top bit off. The only programs requiring modification were those that set the top (leftmost) bit of a word containing an address. This also affected address comparisons: The leftmost bit of a word is also interpreted as a sign-bit in 2's complement arithmetic, indicating a negative number if bit 0 is on. Programs that use signed arithmetic comparison instructions could get reversed results. Two equivalent addresses could be compared as non-equal if one of them had the sign bit turned on even if the remaining bits were identical. Most of this was invisible to programmers using high-level languages like COBOL[9] or FORTRAN,[3][10] and IBM aided the transition with dual mode hardware for a period of time.

Certain machine instructions in this 31-bit addressing mode alter the addressing mode bit as a possibly intentional side effect. For example, the original subroutine call instructions BAL, Branch and Link, and its register-register equivalent, BALR, Branch and Link Register, store certain status information, the instruction length code,[11] the condition code and the program mask, in the top byte of the return address. A BAS, Branch and Store, instruction was added to allow 31-bit return addresses. BAS, and its register-register equivalent, BASR, Branch and Store Register, was part of the instruction set of the System/360 Model 67, which was the only System/360 model to allow addresses longer than 24 bits. These instructions were maintained, but were modified and extended for 31-bit addressing.

Additional instructions in support of 24/31-bit addressing include two new register-register call/return instructions which also effect an addressing mode change (e.g. Branch and Save and Set Mode, BASSM,[12] the 24/31 bit version of a call where the linkage address including the mode is saved and a branch is taken to an address in a possibly different mode, and BSM, Branch and Set Mode, the 24/31 bit version of a return, where the return is directly to the previously saved linkage address and in its previous mode). Taken together, BASSM and BSM allow 24-bit calls to 31-bit (and return to 24-bit), 31-bit calls to 24-bit (and return to 31-bit), 24-bi calls to 24-bit (and return to 24-bit) and 31-bit calls to 31-bit (and return to 31-bit).

Like BALR 14,15 (the 24-bit-only form of a call), BASSM is used as BASSM 14,15, where the linkage address and mode are saved in register 14, and a branch is taken to the subroutine address and mode specified in register 15. Somewhat similarly to BCR 15,14 (the 24-bit-only form of an unconditional return), BSM is used as BSM 0,14, where 0 indicates that the current mode is not saved (th program is leaving the subroutine, anyway), and a return to the caller at the address and mode specified in register 14 is to be taken. Refer to IBM publication MVS/Extended Architecture System Programming Library: 31-Bit Addressing, GC28-1158-1, for extensive examples of the use of BAS, BASR, BASSM and BSM, in particular, pp. 29–30.

370/ESA architecture

In the 1990s IBM introduced 370/ESA architecture (later named 390/ESA and finally ESA/390 or System/390, in short S/390), completing the evolution to full 31-bit virtual addressing and keeping this addressing mode flag. These later architectures allow more than 2 GB of physical memory and allow multiple concurrent address spaces up to 2 GB each in size. As of mid-2006 there were too many programs unduly constrained by this multiple 31-bit addressing mode.

z/Architecture

IBM broke the 2 GB linear addressing barrier ("the bar") in 2000 with the introduction of the first 64-bit z/Architecture system, the IBM zSeries Model 900.[1][13] Unlike the XA transition, z/Architecture does not reserve a top bit to identify earlier code. z/Architecture maintains compatibility with 24-bit and 31-bit code, even older code running concurrently with newer 64-bit code.

Linux/390

Since Linux/390 was first released for the existing 32-bit data/31-bit addressing hardware in 1999, initial mainframe Linux applications compiled in pre-z/Architecture mode are also limited to 31-bit addressing. This limitation disappeared with 64-bit hardware, 64-bit Linux on z Systems, and 64-bit Linux applications. The 64-bit Linux distributions still run 32-bit data/31-bit addressing programs. IBM's 31-bit addressing allows 31-bit code to make use of additional memory. However, at any one instant, a maximum of 2 GB is in each working address space. For non-64-bit Linux on processors with 31-bit addressing, it is possible to assign memory above the 2 GB bar as a RAM disk. 31-bit Linux kernel (not userspace) support was removed in version 4.1.[14]

References

1. "A brief history of virtual storage and 64-bit addressability" (https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconcepts_102.htm)
2. "with transitional support for 26-bit"
3. KE Plambeck (2002). "Development and attributes of z/Architecture" (https://www.cl.cam.ac.uk/teaching/0607/CompArch/ibm-z-plambeck.pdf) (PDF).
4. Robert T. Fertig (May 1983). "XA: The View From The Trenches (pp.122-136)". *Datamation*.
5. Ronald L. Bond (May 1983). "XA: The View From White Plains (pp.139-152)". *Datamation*.

6. "...to run in the 31-bit area above the line,..." "Rewriting to run in 31 bit area" (https://books.google.com/books?id=nI091j0HhhwC). *Computerworld*. October 27, 1986. p. 13.
7. "... the high order bit in the last fullword must be set to one to indicate the end of the list." "WAIT — Wait for one or more events" (https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ieaa400/wait.htm).
8. Indeed, in a variable length parameter list of addresses, the last address entry traditionally had its most significant bi set to 1, whereas the other address entries were required to have their most significant bit set to 0.
9. "VS Cobol II compiles Cobol programs in 31-bit" (https://books.google.com/books?id=hSBrPSYgjI4C)
10. "to accommodate large arrays in FORTRAN."
11. Because the instruction length code is 00b for a BALR and is 01b for a BAL, the high order bit is always guaranteed to be set to 0, thereby indicating 24-bit mode, for BALR and BAL on XA and later systems.
12. "BASSM (branch and save and set mode)" "Using the BASSM and BSM instructions" (https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ieaa600/iea3a6_Using_the_BASSM_and_BSM_instructions.htm).
13. "2-gigabyte address from the user private area is called the bar" "Introduction to the New Mainframe: z/OS Basics" (ftp://ftp.networking.ibm.com/systems/z/z_Redbooks/sg246366_student.pdf) (PDF).
14. "4.1 Merge window, part 1" (https://lwn.net/Articles/640297/). LWN. April 15, 2015.

Retrieved from "https://en.wikipedia.org/w/index.php?title=31-bit&oldid=861210607"