The number of cycles of the innermost comparison loop is close to the theoretical minimum, and the loop may be made very fast. The amount of data movement within the store is kept within very reasonable bounds. Quicksort is therefore likely to recommend itself as the standard sorting method on most computers with a large enough random-access store to make internal sorting worth while.

Acknowledgement

This paper is published by kind permission of Elliott Brothers (London) Ltd.

Reference


Zero-Address Computers

By P. Wegner

The literature on digital computers makes a distinction between one-address, two-address and three-address machines, where the nature of the beast is determined by the number of references to the main random access memory permitted in a single instruction of the basic machine code. After some initial confusion, the one-address machine emerged as the dominant type, and its pre-eminent position has been largely unchallenged during the past five years or so.

I should like to draw attention to the fact that the position of the one-address machine is being challenged by a new type of animal which, on the basis of the above classification, must be called a zero-address machine.

The basic arithmetic operations are of the three-operand type. For instance, the operation \( C := A - B \) has two operands as input and one operand as output. Three-address machines permit basic machine instructions to refer to the three operands explicitly. Two-address machines refer to two operands explicitly, and one operand (usually the result) implicitly. One-address machines refer to one operand explicitly, and assume that an arithmetic operation, such as ADD, finds its second operand in an independently specified register and stores its result in a second, possibly identical, independently specified register.

A zero-address instruction can be defined as one where the location of all relevant operands is specified by convention, so that no operand need be designated explicitly. Zero-address logical and arithmetic operations are available in computers like the English Electric KDF 9 or the Burroughs B 5000, in which all operands required as input to an arithmetic operation are previously stored in a group of temporary storage registers with last-in-first-out properties, variously known as a nesting store, a stack or a pushdown store. Furthermore, the result of an arithmetic operation on operands in the pushdown store is left in a register in the pushdown store from which it may immediately be used for further zero-address arithmetic operations. For instance, three-address operations, such as addition, perform the addition operation on the two top registers of the pushdown store, reduce the size of the pushdown store by deleting the top register, and store the result in the new top register (previously the second register), where it is available for immediate use for subsequent computation.

By means of a pushdown store it is possible to specify arithmetic and logical operations without explicit reference to an operand. However, a machine which is truly a zero-address machine, requires elimination of references to operands for all operations, including data transmission operations typified by FETCH and STORE. This is accomplished in the Burroughs B 5000 by channelling all references to operands through an operand directory known as the "Program Reference Table." Since grouped data, such as arrays, need be specified only by a single "data descriptor" in such a directory, the number of bits required to reference such a directory will be smaller than the number of bits required to reference the memory as a whole. An indirect addressing technique of this kind eliminates the need to refer to operands explicitly in terms of the memory location which they occupy, so that data transmission instructions of this kind may, in some sense, be regarded as zero-address instructions.

The principal remaining class of memory-address references is that of labels. Labelling, and transfers to labels within a program, may also be dealt with by a program directory technique. A machine like the Burroughs B 5000, which references operands and labels through a program directory, may therefore be regarded as a zero-address computer.

Zero-address computers are more economical in their utilization of memory space for programs than one-address computers, since it is unnecessary to provide space in an instruction for referencing a memory location. For instance, the B 5000 has 12-bit instructions and permits four instructions to a computer word.

Furthermore, the fact that arithmetic and logical operations in such an instruction code are "pure" operations, unencumbered by operands, gives rise to a closer correspondence between mathematical source language and basic machine code than is the case in one-address computers. The construct \( x := y \) in a mathematical source language has a precise counterpart in the target language; and, in general, source language constructs retain their identity in the target language, although the order of their appearance may be changed. This correspondence leads to simpler and faster translation programs than in the case of one-address machines.

To sum up, there are three principal advantages in a zero-address basic machine code:

1. Machine language instructions can be short, since no explicit reference to operands in a large random-access memory is required.
2. The structure of the machine language is close to the structure of mathematical source languages, leading to fast translation procedures.
3. Execution of sequences of arithmetic and logical operations is speeded up since the number of references to random-access memory is reduced.

In view of these advantages, computers with zero-address arithmetic and logical machine instructions operating through a pushdown store have probably come to stay. The case for zero-address operations for all classes of basic machine instructions is not quite as compelling. However, a consistent zero-address instruction code, permitting only indirect references to locations in the random-access memory, has a great deal to be said for it.