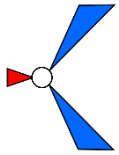
1588 CA1

bncdoc.id	EUS
bncdoc.author	Garside, R G
bncdoc.year	1980
bncdoc.title	The architecture of digital computers.
bncdoc.info	The architecture of digital computers. Sample containing about 40516 words from a book (domain: applied science)
Text availability	Ownership has not been claimed
Publication date	1975-1984
Text type	Written books and periodicals
David Lee's classification	W_ac_tech_engin

<1588/c>



Key:
Footprint ConEn1
Footprint ConEn2
Footprint ConEn3

the 3 representations for negative numbers could be used, sign-and-magnitude being most common. All the usual arithmetic operations of addition, subtraction, multiplication, division, and comparison are provided, as in 2.1. Only decimal arithmetic may be provided, or separate instruction sets might treat each computer word as either a binary or a decimal value. A few of these computers also have a decimal floating-point format. Here the fraction is held as a series of decimal digits, with a floating. point base of ten rather than a power of the 3 representations for negative numbers could be used, sign-and-magnitude being most common. All the usual arithmetic operations of addition, subtraction, multiplication, division, and comparison are provided, as in 2.1. Only decimal arithmetic may be provided, or separate instruction sets might treat each computer word as either a binary or a decimal value. A few of these computers also have a decimal floating-point format. Here the fraction is held as a series of decimal digits, with a floating. point base of ten rather than a power of two; normally the exponent is still held in binary format. Few modern large or medium.sized computers have such word-oriented decimal arithmetic. It is more usual nowadays to provide decimal arithmetic (if at all) on a data.type derived from character string handling. It is to such character string formats that we now turn. Character-oriented computers and the IBM 1401 The word-oriented architecture derives from viewing computers as calculating devices. Large numbers of calculations are to be done, so that the design aim is to choose a word size which gives suitable precision, and to do all arithmetic in the most efficient way, using fixed or floating.point binary format. On this view transput will be relatively infrequent, so that the cost of converting to and from decimal format is acceptable. However, in commercial applications the emphasis is different, as a glance at the Cobol language indicates. The basic unit is the (variable-length) string of characters. The manipulations performed may include arithmetic operations on some of these character strings, but we will probably wish to do them directly on the decimal digits, rather than convert to and from binary for each arithmetic operation. We therefore consider building an architecture round character string manipulation, rather than round word manipulation. The first common computer using this character.oriented architecture was the (second.generation) IBM 1401 (McCracken 1962, Bell and Newell 1971, pp. 225-34). The basic addressable unit of storage on the IBM 1401 computer is the character, which holds eight bits. Six data bits encode **a character set** consisting of **the digits 0 to 9**, **the letters A to Z**, and

a number of special symbols

(such as **blank and comma**), as shown in Figure 2.19. The seventh bit is a parity bit, and the eighth is known as the word mark bit; by the usual definition, this computer therefore has a 7-bit word. Each operand is a string of one or more consecutive characters. In order to facilitate arithmetic, an operand is specified by the address of its right-hand (least-significant) character, and the left-hand (most-significant)

	character of the operand is indicated by having its word mark bit set to one. Because operands may be very long, no accumulator is provided on this computer; all operations take place between operands in the store. The most basic instruction is the move, which copies characters from a source field to a destination, until a character with the word mark bit set is encountered in one of the two fields. For example, suppose we had two fields (or character strings) in the store at addresses 300 and 350, as shown in Figure 2.20 (a); the addresses are those of the right.hand characters of the fields, and the characters whose word mark bits are set are indicated by underlining. Then if the computer executes the instruction, Move Characters to Word Mark: Source Address 300: Destination Address 350 the result will be as shown in Figure 2.20 (b). Characters have been copied from the first field to the second, until a word mark (here in the source field) is encountered. This instruction does not affect the pattern of word marks in the destination field, but there are other instructions to set or clear the word mark bit in a designated character, to clear all word marks in a designated area, or to copy a source string and word mark to a designated area (ignoring and clearing any word mark: bits in the destination field encountered in the process). To describe how arithmetic is performed on the IBM 1401, we must look at the representation of characters a little more closely. The six data bits of a character are divided into two zone bits and four numeric bits. All six bits are used to represent most symbols in the character set, but it can be seen from Figure 2.19 that the digits 0 to 9 are represented by the four numeric digits, with the zone bits set to " 00 ". However, the zone bits of the right-hand (least significant) digit of a numeric field are set to " 10 " for a negative value and to any other pattern (00,01, or 1 I, but usually 1 I) for a positive value. Thus numeric operands are held in sign-and-magnitude form, with the sign on the right, since operands are processed from right to left. As an example we see from figure 2.19 that 103 would be represented in a four-character field (ignoring word mark and parity bits) as A consequence of this technique of representing negative values is that some bit patterns have two interpretations; thus " 100011 " represents the character " L " as well as 3 and a " minus " sign. Optional or standard instructions are provided for the four arithmetic operations. For example, the " add