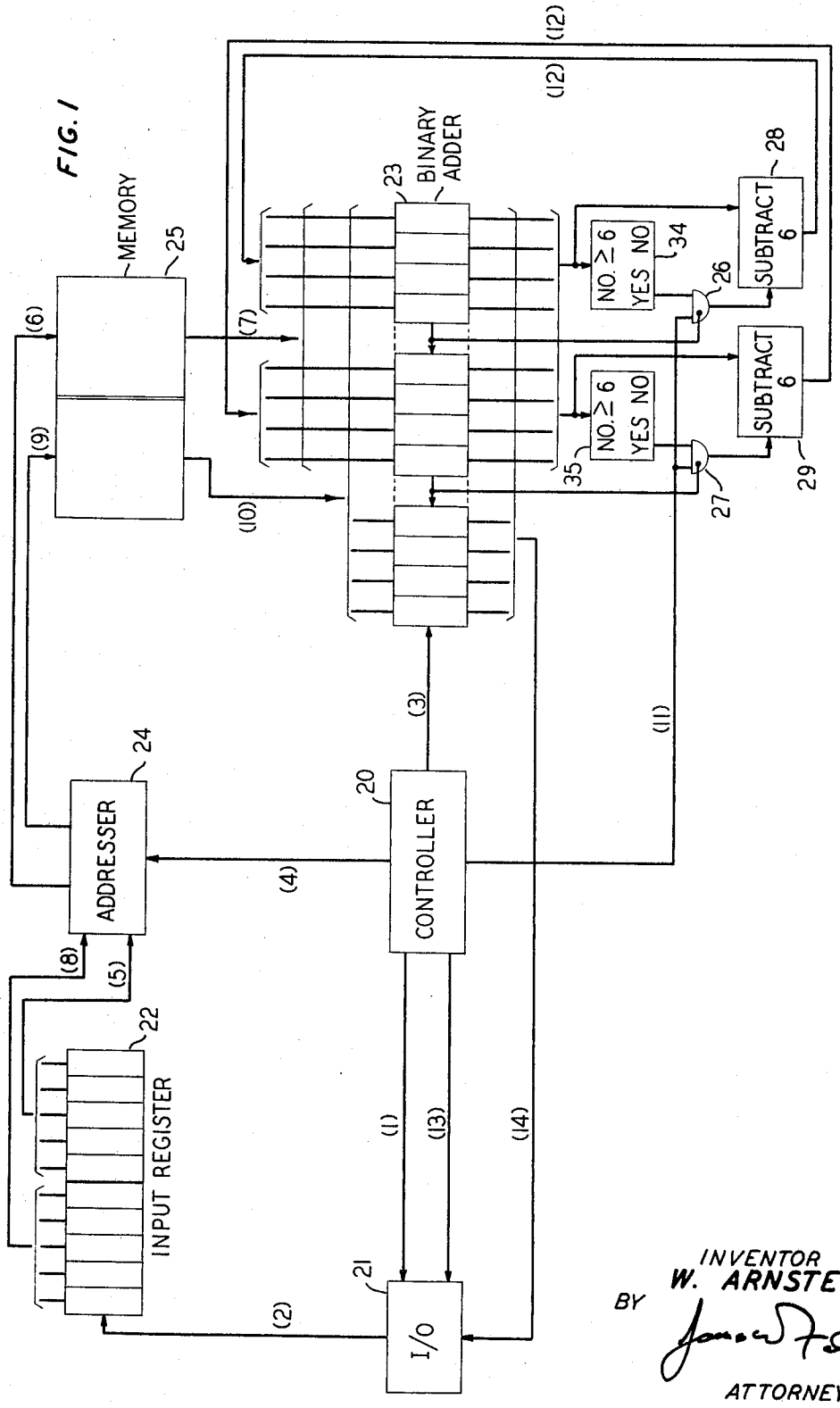
April 28, 1970 W. ARNSTEIN 3,509,328

CODE CONVERSION

Filed March 15, 1965 5 Sheets-Sheet 1

FIG. 1



INVENTOR
W. ARNSTEIN

BY

ATTORNEY

April 28, 1970 W. ARNSTEIN 3,509,328

CODE CONVERSION

Filed March 15, 1965 5 Sheets-Sheet 2

FIG. 2

ADDRESS	CONTENTS	ADDRESS	CONTENTS
00000 (0)	0000,0011,0011 (0,3,3)	00000 (0)	0011,0011 (3,3)
00001 (1)	0000,0110,0101 (0,6,5)	00001 (1)	0011,0100 (3,4)
00010 (2)	0000,1001,0111 (0,9,7)	00010 (2)	0011,0101 (3,5)
00011 (3)	0000,1100,1001 (0,12,9)	00011 (3)	0011,0110 (3,6)
00100 (4)	0001,0101,1011 (1,5,11)	00100 (4)	0011,0111 (3,7)
00101 (5)	0001,1001,0011 (1,9,3)	00101 (5)	0011,1000 (3,8)
00110 (6)	0001,1100,0101 (1,12,5)	00110 (6)	0011,1001 (3,9)
00111 (7)	0010,0101,0111 (2,5,7)	00111 (7)	0011,1010 (3,10)
01000 (8)	0010,1000,1001 (2,8,9)	01000 (8)	0011,1011 (3,11)
01001 (9)	0010,1011,1011 (2,11,11)	01001 (9)	0011,1100 (3,12)
01010 (10)	0011,0101,0011 (3,5,3)	01010 (10)	0100,0011 (4,3)
01011 (11)	0011,1000,0101 (3,8,5)	01011 (11)	0100,0100 (4,4)
01100 (12)	0011,1011,0111 (3,11,7)	01100 (12)	0100,0101 (4,5)
01101 (13)	0100,0100,1001 (4,4,9)	01101 (13)	0100,0110 (4,6)
01110 (14)	0100,0111,1011 (4,7,11)	01110 (14)	0100,0111 (4,7)
01111 (15)	0100,1011,0011 (4,11,3)	01111 (15)	0100,1000 (4,8)
10000 (16)	0101,0100,0101 (5,4,5)	10000 (16)	0100,1001 (4,9)
10001 (17)	0101,0111,0111 (5,7,7)	10001 (17)	0100,1010 (4,10)
10010 (18)	0101,1010,1001 (5,10,9)	10010 (18)	0100,1011 (4,11)
10011 (19)	0110,0011,1011 (6,3,11)	10011 (19)	0100,1100 (4,12)
10100 (20)	0110,0111,0001 (6,7,3)	10100 (20)	0101,0011 (5,3)
10101 (21)	0110,1010,0101 (6,10,5)	10101 (21)	0101,0100 (5,4)
10110 (22)	0111,0011,0111 (7,3,7)	10110 (22)	0101,0101 (5,5)
10111 (23)	0111,0110,1001 (7,6,9)	10111 (23)	0101,0110 (5,6)
11000 (24)	0111,1001,1011 (7,9,11)	11000 (24)	0101,0111 (5,7)
11001 (25)	1000,0011,0011 (8,3,3)	11001 (25)	0101,1000 (5,8)
11010 (26)	1000,0110,0101 (8,6,5)	11010 (26)	0101,1001 (5,9)
11011 (27)	1000,1001,0111 (8,9,7)	11011 (27)	0101,1010 (5,10)
11100 (28)	1000,1100,1001 (8,12,9)	11100 (28)	0101,1011 (5,11)
11101 (29)	1001,0101,1011 (9,5,11)	11101 (29)	0101,1100 (5,12)
11110 (30)	1001,1001,0011 (9,9,3)	11110 (30)	0110,0011 (6,3)
11111 (31)	1001,1100,0101 (9,12,5)	11111 (31)	0110,0100 (6,4)

FIG. 4

ADDRESS	CONTENTS
00000 (0)	0011,0011,0000,0011,0011 (3,3,0,3,3)
00001 (1)	0011,0100,0000,0110,0101 (3,4,0,6,5)
00010 (2)	0011,0101,0000,1001,0111 (3,5,0,9,7)
00011 (3)	0011,0110,0000,1100,1001 (3,6,0,12,9)
00100 (4)	0011,0111,0001,0101,1011 (3,7,1,5,11)
00101 (5)	0011,1000,0001,1001,0011 (3,8,1,9,3)
00110 (6)	0011,1001,0011,1100,0101 (3,9,1,12,5)
00111 (7)	0011,1010,0010,0101,0111 (3,10,2,5,7)
01000 (8)	0011,1011,0010,1000,1001 (3,11,2,8,9)
01001 (9)	0011,1100,0010,1011,1011 (3,12,2,11,11)
01010 (10)	0100,0011,0011,0101,0011 (4,3,3,5,3)
01011 (11)	0100,0100,0011,1000,0101 (4,4,3,8,5)
01100 (12)	0100,0101,0011,1011,0111 (4,5,3,11,7)
01101 (13)	0100,0110,0100,0100,1001 (4,6,4,4,9)
01110 (14)	0100,0111,0100,0111,1011 (4,7,4,7,11)
01111 (15)	0100,1000,0100,1011,0011 (4,8,4,11,3)
10000 (16)	0100,1001,0101,0100,0101 (4,9,5,4,5)
10001 (17)	0100,1010,0101,0111,0111 (4,10,5,7,7)
10010 (18)	0100,1011,0101,1010,1001 (4,11,5,10,9)
10011 (19)	0100,1100,0110,0011,1011 (4,12,6,3,11)
10100 (20)	0101,0011,0110,0111,0011 (5,3,6,7,3)
10101 (21)	0101,0100,0110,1010,0101 (5,4,6,10,5)
10110 (22)	0101,0101,0111,0011,0111 (5,5,7,3,7)
10111 (23)	0101,0110,0111,0110,1001 (5,6,7,6,9)
11000 (24)	0101,0111,0111,1001,1011 (5,7,7,9,11)
11001 (25)	0101,1000,1000,0011,0011 (5,8,8,3,3)
11010 (26)	0101,1001,1000,0110,0101 (5,9,8,6,5)
11011 (27)	0101,1010,1000,1001,0111 (5,10,8,9,7)
11100 (28)	0101,1011,1000,1100,1001 (5,11,8,12,9)
11101 (29)	0101,1100,1001,0101,1011 (5,12,9,5,11)
11110 (30)	0110,0011,1001,1001,0011 (6,3,9,9,3)
11111 (31)	0110,0100,1001,1100,0101 (6,4,9,12,5)

FIG. 3

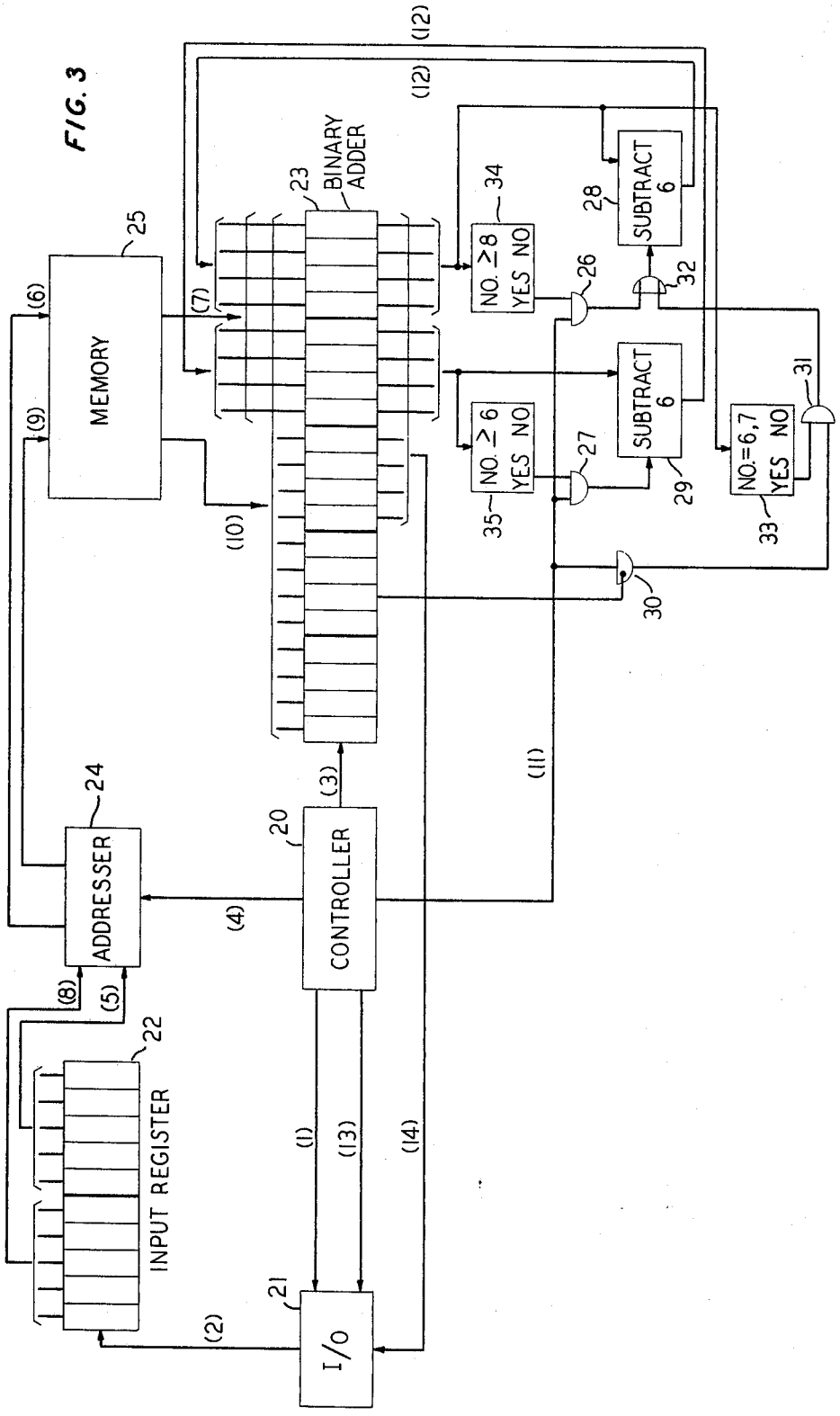

(12)
(12)
23
BINARY ADDER
34
28
SUBTRACT 6
25
NO. ≥8
YES NO
26
32
(6)
(7)
MEMORY
SUBTRACT 6
31
(9)
NO. ≥ 6
YES NO
27
NO.=6,7
YES NO
(10)
35
29
33
30
(11)
24
(3)
20
ADDRESSER
CONTROLLER
(4)
(8)
(5)
22
INPUT REGISTER
(1)
(13)
(14)
21
(2)
I/O

FIG. 5

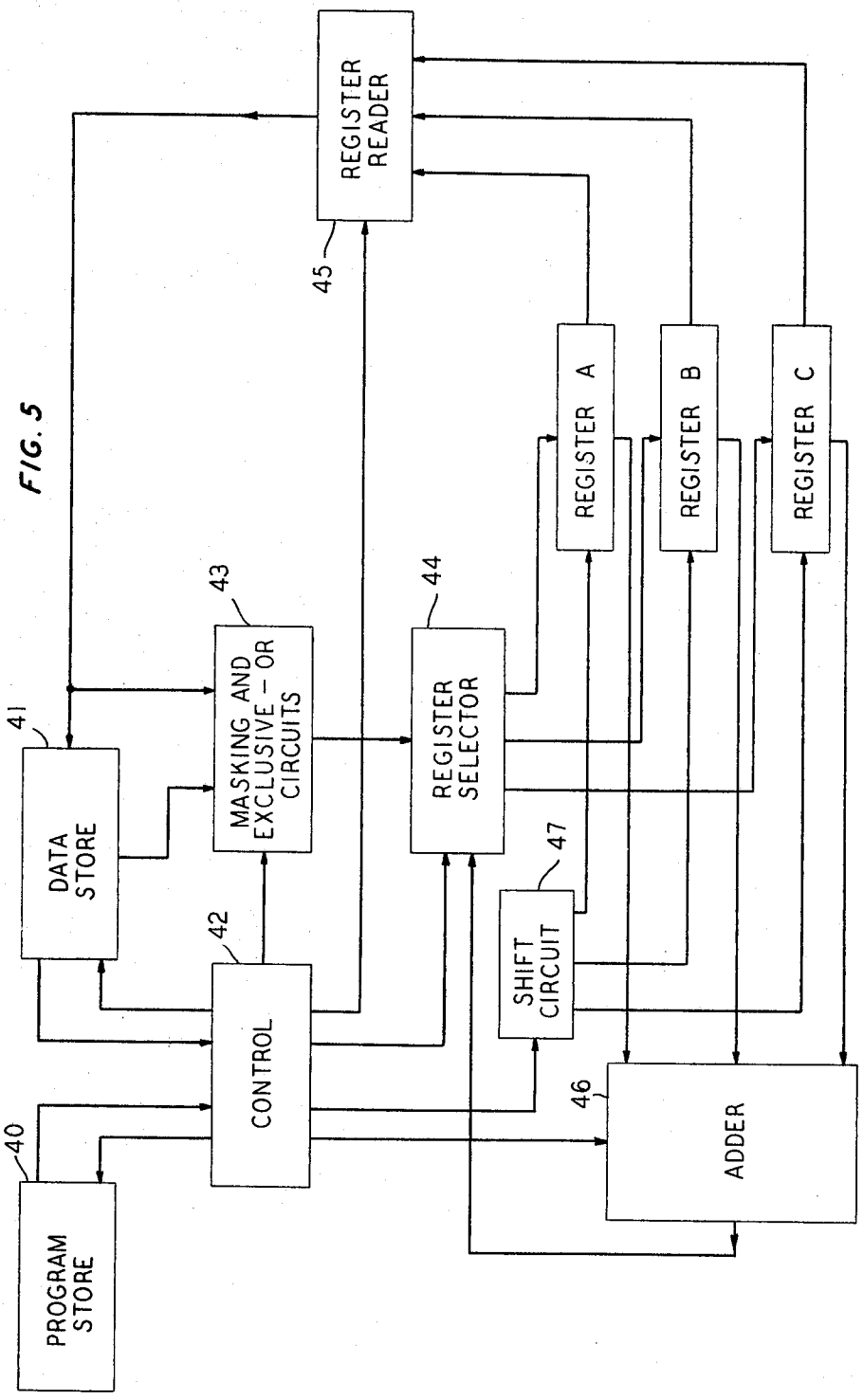

REGISTER READER
45
REGISTER A
REGISTER B
REGISTER C
43
44
41
MASKING AND EXCLUSIVE - OR CIRCUITS
REGISTER SELECTOR
DATA STORE
47
SHIFT CIRCUIT
42
CONTROL
46
ADDER
40
PROGRAM STORE

FIG. 7

ADDRESS	CONTENTS	
N + 0000	11000,11000	[0,0]
N + 0001	00011,00011	[1,1]
N + 0010	00101,00101	[2,2]
N + 0011	00110,00110	[3,3]
N + 0100	01001,01001	[4,4]
N + 0101	01010,01010	[5,5]
N + 0110	01100,11000	[6,0]
N + 0111	10001,00011	[7,1]
N + 1000	00101,00101	[2,2]
N + 1001	00110,00110	[3,3]
N + 1010	01001,01001	[4,4]
N + 1011	01010,01010	[5,5]
N + 1100	01100,01100	[6,6]
N + 1101	10001,10001	[7,7]
N + 1110	10010,10010	[8,8]
N + 1111	10100,10100	[9,9]

FIG. 6

ADDRESS	CONTENTS
N + 0000	0000,0000 (0,0)
N + 0001	0000,0000 (0,0)
N + 0010	0000,0000 (0,0)
N + 0011	0000,0000 (0,0)
N + 0100	0000,0000 (0,0)
N + 0101	0000,0000 (0,0)
N + 0110	0110,0110 (6,6)
N + 0111	0110,0000 (6,0)
N + 1000	1010,1010 (10,10)
N + 1001	1010,1010 (10,10)
N + 1010	1110,1110 (14,14)
N + 1011	1110,1110 (14,14)
N + 1100	1010,1010 (10,10)
N + 1101	1010,1010 (10,10)
N + 1110	0110,0110 (6,6)
N + 1111	0110,0110 (6,6)

United States Patent Office

3,509,328

Patented Apr. 28, 1970

3,509,328

CODE CONVERSION

Walter Arnstein, Matawan, N.J., assignor to Bell Telephone Laboratories, Incorporated, New York, N.Y., a corporation of New York

Filed Mar. 15, 1965, Ser. No. 439,633

Int. Cl. G06f 5/02; H04l 3/00

U.S. Cl. 235—155

3 Claims

ABSTRACT OF THE DISCLOSURE

An arrangement is disclosed for converting binary numbers to binary coded decimal numbers through the use of data processing apparatus. The binary number to be converted is registered, divided into two portions and the bits comprising each portion are employed as an address to access a respective binary coded decimal memory word at least some of whose digits are stored in the excess-3 code. The memory words corresponding to the two portions of the binary number to be converted are added in a conventional binary adder and the digits of the sum are corrected by subtracting the digit 6 when required; i.e., when the sum does not produce a "carry" to the next higher digit. The presence or absence of a "carry" may be determined by examining a digit position in the sum lying outside the range of the digits required to indicate the sum.

This invention relates to data processors and more particularly to organizations for use therein to convert data in one code to another code.

In most data processors data is stored in bit form with each bit being either a 1 or a 0. Often, however, the data which is to be operated upon is not in binary form. Consider, for example, data stored in the binary coded decimal (BCD) form. Each decimal digit requires four bits, the four bits representing the binary value of the respective digit. For example, the BCD equivalent of the decimal number 87 is 1000 0111. If the input to the data processor and its output are both in BCD form the data stored in the machine is often in the same code.

Storing BCD numbers however may be highly inefficient. For example, consider a machine in which each stored data word contains 20 bits. Each memory location may represent any one of 2^{20} data words, i.e., any one of 1,048,576 numbers. The same memory location can store only one of 10^5, or 100,000, numbers in BCD form. Thus storing BCD numbers in the memory may be very wasteful. This is due to the fact that in the BCD code six out of the 16 possible four-bit "digits" are not permissible combinations.

The present invention may be illustrated by considering the number of memory bits required to store the decimal numbers 000 through 999. These numbers may have to be stored for example if a binary number derived in the machine must be translated into its BCD equivalent. Each of the binary numbers 0000000000 through 1111100111 (decimal 999) is associated with one of the stored numbers in the BCD "dictionary." To store each number in BCD 12 bits are required and thus 12,000 bits must be used to store the entire range of values.

It is a principal object of this invention to enable relatively few bits to represent the data required to convert a binary number to its BCD equivalent, with the processing capabilities of the processor being used to manipulate the stored data to derive the required BCD number.

In the illustrative embodiments of my invention 1,000 different 12-bit words are not stored in the memory. Instead, two tables are provided in the memory, each containing 32 words. The machine operates on binary words.

(If the input is in BCD there are well-known methods for converting the input data to binary form.) When it is desired to derive the BCD equivalent of a binary number one word is extracted from each of the two tables, the two words are added together, and after a further manipulation the result is in the required BCD form. One of the tables contains 32 12-bit numbers and the other contains 32 eight-bit numbers. Thus, only 640 bits of tabular data are required in the memory to derive the BCD equivalent of any 10-bit binary number where the value of the binary number lies within the range 000 through 999. In the prior art 12,000 bits have been required for the memory "dictionary" which serves as the basis for converting the 10-bit binary numbers to their BCD equivalents.

Consider a typical 10-bit binary number representing one of the values 000 through 999. If this number is split into two five-bit binary segments A and B, the value of the entire binary number may be expressed as $32A+B$, where A and B are both less than 32. The first table contains the BCD equivalents of the numbers 0 through 31. The second table contains the BCD equivalents of the numbers 32×0 through 32×31. The first five bits (B) in the binary member to be converted are examined and the respective eight-bit BCD number in the first table is transmitted to an adder. The last five bits (A) in the binary number to be converted are examined and the respective 12-bit BCD number in the second table is transmitted to the adder. The adder then derives the sum of the two numbers, the result being the BCD equivalent (12 bits) of the 10-bit binary number to be converted having one of the values 000 through 999.

Using two tables and an adder in this manner rather than one table of much greater length enables the machine "dictionary" to be much shorter than that which would otherwise be required. However, the numbers in the two tables are stored in BCD form and it would appear that a BCD adder would be required to add them. Since many machines operate on binary words they are provided with binary adders and two BCD numbers may not be added correctly by a binary adder. For example, consider the addition of the decimal numbers 748 and 123. The result of the addition of the two BCD equivalents on a binary adder is as follows:

7 4 8 1 2 3	0111 0001	0100 0010	1000 0011
8 7 1	1000	0110	1011

The first BCD digit is 8 as required. The second, however, is 6 rather than 7, and the last digit does not even have any meaning in the BCD code since the range of BCD values is limited to 0000 through 1001. Upon examination of the example above it is seen that the difficulty is two-fold. Frist, two BCD digits when added together may exceed the value 1001 on a binary added. Second, if the sum exceeds 1001 but is less than 1111 a carry will not be generated to the next higher position.

In order to practice the invention in a machine equipped only with a binary adder, certain BCD digits in the two tables are represented in the excess-3 code, whose digits are greater by 3 than their ordinary BCD counterparts. For example, the number 8 (BCD 1000) is represented in excess-3 by 1011. Two numbers in the excess-3 code may be added on a binary added to yield a BCD sum if the result is modified as follows: The number 6 (binary 0110) is subtracted from any digit in the sum if the digit is equal to or greater than 6 and a carry was not generated during the addition to the next position of higher significance. Consider the example given above.

The BCD digits are now in the excess-3 code and the sum derived by a binary adder is as follows:

7 4 8	1010	0111	1011
1 2 3	0100	0101	0110
8 7 1	1110	1101	0001

The first digit in the sum is 1110. This number is greater than 6 and when 6 is subtracted from it the resulting value, 1000, is the correct one. The second digit, 1101, is also greater than 6 and no carry is generated from the second position to the third position during the addition. When the binary number 0110 is subtracted from the binary number 1101 the result is 0111, or decimal 7 as required. The last digit derived during the addition operation is 0001. This number is less than decimal 6 and is left unchanged. As seen it is the required result.

It will be noted that the rule given above requires that binary 0110 not be subtracted from a resulting dibit even if the digit is greater than or equal to decimal 6 if a carry was generated during the addition process. This may be illustrated by the following example in which the second number is 129 rather than 123.

7 4 8	1010	0111	1011
1 2 9	0100	0101	1100
8 7 7	1110	1101	0111

The first two BCD digits 1110 and 1101 are the same as those derived in the previous example and the correct values are obtained when 0110 is subtracted from each of them. The last BCD digit in the result is 01111. This number is the BCD equivalent of decimal 7 and ordinarily 0110 should be substracted from it. However, in adding the two excess-3 numbers on a binary added a carry is generated from the rightmost group ("units") of four bits to its higher-order neighbor ("tens"). Since a carry is generated the number 0110 is not subtracted from the first BCD digit even though it is greater than 6. The result of the addition, 0111, decimal 7, is the correct one since the least significant digit in the decimal sum is 7. In fact, if a carry is not generated the sum digit must be greater than or equal to 6 since each excess-3 digit is at least 3. Consequently, the rule given above for modifying the sum digits may be simplified: The number 6 (binary 0110) is subtracted from any digit in the sum if a carry was not generated during the addition to the next position of higher significance.

Thus by storing the numbers in the two tables in the excess-3 code a binary adder may be used to derive a BCD sum, i.e., to derive the BCD equivalent of the initial binary number to be converted. However, to derive the final result it may be necessary to subtract the binary number 0110 from each BCD digit in the sum. It is a simple matter to examine each digit to determine if it is greater than or equal to 6. However, a digit in the sum even if it is greater than or equal to 6 must not be "corrected" if a carry was generated from the respective position during the addition operation. It is therefore necessary to determine whether a carry is generated from binary position 4 to binary position 5, from binary position 8 to binary position 9, etc. Many data processors are not provided with the required carry detectors.

It is another object of my invention, in certain embodiments thereof, to practice my code conversion method even in a machine not equipped with carry detection facilities.

In these certain embodiments of the invention the two tables are actually combined into one table having 32 20-bit words. The eight most significant bits in each word represent the excess-3 equivalent of the respective value of B in 32A+B, namely, the excess-3 encoding of one of the binary numbers 00000 through 11111. The 12 least significant bits in each word, similarly represent the excess-3 encoding of the respective value of 32A in

32A+B. (As will be seen below it is not necessary that the middle four bits in each word be in the excess-3 code). The memory includes 20-bit words and thus the system is provided with a 20-bit adder. The binary number to be converted to BCD is divided into two five-bit parts, A and B. The eight most significant bits in the word in the table associated with the binary number B are transmitted to the eight least significant positions of the adder. For the binary word A it is necessary to transmit the 12 least significant bits in the respective table word. However, the entire 20-bit word is transmitted to the adder. The adder derives the sum of the eight-bit word and the 20-bit word transmitted to it. The BCD result eventually appears in the 12 least significant positions. To correct the result following the rule given above it is necessary to determine whether a carry was generated during the addition from position 4 to position 5 and from position 8 to position 9. In the illustrative embodiments of the invention, as will become apparent below carries generated from position 8 to position 9 need not be detected. Although a carry detector is not provided for determining whether a carry was generated from the first excess-3 digit group to the second, due to the organization of the table and the fact that the second word transmitted to the adder is a full 20-bit word rather than a 12-bit word, the fifteenth bit in the adder sum may be examined and the value of this bit may be used as an indication of whether the rightmost four-bit digit in the result should be corrected. Thus although only the 12 least significant bits in the adder output are required for the BCD result the fifteenth bit may be examined to determine whether a carry was generated. In general, it is possible to organize the "dictionary" and to transmit data to the adder in such a manner that unused bits in the adder output may be examined to determine which digit positions must be corrected.

It is a feature of this invention to represent the BCD equivalents of r-bit binary numbers in two tables with the BCD equivalents of the binary numbers 0 through 2^n-1 being stored in the first table and the BCD equivalents of the binary numbers $2^n(0)$ through $2^n(2^m-1)$ being stored in the second table, where $m+n=r$.

It is another feature of this invention to store some of the BCD numbers in the two tables in the excess-3 code.

It is another feature of this invention to add the word in the first table associated with the first n bits of the binary number of be converted to the word in the second table associated with the last m bits in the binary number to be converted to derive the BCD equivalent in the "excess-6" code of the binary number to be converted.

It is another feature of this invention to correct the BCD digits in the derived sum by subtracting 0110 from each digit if the digit is 0110 or greater and a carry was not generated to the next higher BCD digit position during the addition.

It is still another feature of this invention, in some embodiments thereof, to determine whether carries were generated from one BCD position to another during the addition by adding additional bits not required for representing the two BCD equivalents of the n and m parts of the binary word to be converted and examining the values of bits in the derived sum which are not in positions required to represent the final BCD value.

While the objects, features and brief description of my invention presented above relate to a binary-to-BCD conversion scheme, as will be shown hereinafter, my method is applicable to other types of conversion, e.g., binary to 2-out-of-5.

Further objects, features and advantages of the invention will become apparent below upon consideration of the following detailed description in conjunction with the drawing in which:

FIG. 1 is a schematic representation of a first illustrative circuit which may be incorporated in a data processor to carry out the method of the invention;

FIG. 2 represents data values stored in the memory of FIG. 1;

FIG. 3 is a schematic representation of a second illustrative circuit which may be incorporated in a data processor to carry out the method of the invention;

FIG. 4 represents data values stored in the memory of FIG. 3;

FIG. 5 is a schematic representation of a general-purpose data processor in which the method of my invention may be practiced; and

FIGS. 6 and 7 represent data values stored in the data store of FIG. 5 which may be used to convert binary numbers to BCD and 2-out-of-5 codes.

The circuit of FIG. 1 is operative in itself to convert a binary number to a BCD number and may be incorporated for this purpose in a large data processor. The various operations are governed by controller **20**. The controller transmits signals to the various other units. The numbers in parentheses associated with the various cables are used for the purpose of showing the sequential flow of information in the circuit.

A 10-bit binary number which lies within the range between decimal 000 and decimal 999 appears in input/output unit **21**. The controller transmits a signal to the input/output unit which in turn stores the 10-bit binary input member in input register **22**. The controller then pulses the reset conductor to reset the binary adder **23**. The addresser **24** is then pulsed to initiate its operation.

Before proceeding with the operation of the addresser it is necessary to examine the data values stored in memory **25**. Referring to FIG. 2 it is seen that the memory is divided in two parts. The right hand portion of the memory consists of 32 eight-bit locations. The addresses of these locations range from 00000 to 11111 (decimal 31). The content of each location is the equivalent of the respective address, in the excess-3 code. Adjacent to each eight-bit number the decimal expression of its value in BCD is shown in parentheses. For example, consider location 10001 (decimal 17). Decimal 17 in the excess-3 code comprises the two numbers 4 and 10. Thus the data stored in location 10001 is 0100 1010.

The left hand portion of the memory also has 32 locations with respective addresses 00000 through 11111. The data stored in each of these locations however is 12 bits in length and is a decimally coded equivalent of the respective address multiplied by 32. Only the eight least significant bits in each location are in the excess-3 code; the leftmost four-bit digit is in the standard BCD code. For example, consider memory location 10001 (decimal 17). The product of 17 and 32 is 544. The four leftmost bits in the location are the standard BCD equivalent of the digit 5, 0101. The next two groups of four bits are each the excess-3 equivalent of a respective digit in the product. The number 4 in the excess-3 code is 7 and thus each of the two least significant four-bit groups in location 10001 contains the excess-3 number 0111. As another example consider location 10011 (decimal 19). The product of 19 and 32 is 608. When the two least significant digits are represented in the excess-3 code the resulting decimal numbers are 6, 3 and 11. Thus the 12-bit mixed number stored in memory location 10011 is 0110 0011 1011.

In the fourth step of the sequence addresser **24** is energized. The addresser then examines the five least significant bits in input register **22**. This five-bit number is treated as an address and in step (6) this five-bit address is transmitted to memory **25**. The memory, in step (7), transmits the respective eight-bit word in its right hand portion to the eight least significant positions in binary adder **23**. The addresser, in step (8), examines the five most significant bits in input register **22** and transmits this address to the left hand portion of memory **25**. The memory, in step (10), transmits the 12-bit data word in the respective location to the 12 stages of the

binary adder. The adder then derives the sum of the eight-bit and 12-bit numbers transmitted to it.

After the sum is derived in the binary adder detectors **34** and **35** operate. Each of these detectors determines whether the respective BCD digit in the adder is equal to or greater than 0110. If it is, one input of the respective one of gates **26** and **27** is energized. The controller, in step (11), energizes the other input of each of these gates. It is to be recalled that the number 0110 is to be subtracted from each BCD excess-6 digit in the adder only if the digit is equal to or greater than 0110 and a carry was not generated to the next highest digit position. Gates **26** and **27** are normally enabled but if in the addition process a carry is generated from position 4 to position 5 gate **26** is inhibited from operating and remains inhibited until after the pulse applied by controller **20** in step (11) is terminated. Similar remarks apply to gate **27** and the carry generated from position 8 to position 9 in the adder.

If one (or both) of gates **26** or **27** operates the respective one of subtractors **28** and **29** is enabled. Each of these units subtracts the number 0110 from the number contained in the respective four stages of the adder, and in step (12) erases the initial digit in the adder and stores the difference in its place. In step (13) the controller energizes the input/output unit **21** which, in step (14), reads out the 12-bit BCD number in the adder, this 12-bit number being the BCD equivalent of the initial 10-bit binary number to be converted.

The method of my invention may be best understood by considering a particular example. Suppose the initial 10-bit binary number to be converted is 1001110111. The five least significant bits (decimal 23) are transmitted to memory **25** in step (6) and the eight-bit BCD excess-3 number 0101 0110 (decimal 56) is transmitted to the adder. In step (9) the five most significant bits (decimal 19) in the initial binary word to be converted are transmitted to the memory. The 12-bit binary number transmitted from the memory to the adder is 0110 0011 1011. (This number is the BCD equivalent of the product of 19 and 32, with the two least significant digits being in the excess-3 code.) The adder then derives the sum of the eight-bit and 12-bit numbers transmitted to it. The sum is 0110 1001 0001, and in the course of the addition a carry is generated from position 4 to position 5.

The decimal equivalent of the initial binary number 1001110111 is 631. The most significant BCD digit in the adder is 0110, or decimal 6. This is the desired result. It will be recalled that the most significant BCD digit stored in the left hand portion of memory **25** is in the ordinary BCD code. The excess-3 code is not required because the four most significant bit stages in adder **23** have transmitted to them only one set of bits. The BCD digit in the four most significant stages in the adder are the same as the original digit transmitted from the table or greater than this value by only 1 if a carry is generated from bit position 8 to bit position 9. Consequently the most significant digit in the sum derived by the adder never needs correction.

The middle digit derived by the adder is 1001, decimal 9. This value is greater than 6 and detector **25** operates. Since gate **27** is not inhibited subtractor **29** operates and subtracts 0110 from the middle digit 1001. The difference 0011 is substituted in the middle four stages of the adder in place of the derived digit 1001. The digit 0011, decimal 3, is the correct value since the middle digit of the decimal equivalent (631) of the initial binary number to be converted is a 3.

The least significant digit in the binary adder is 0001. This digit is less than 6 and consequently detector **24** does not operate. It should be noted that even if the digit were greater than 6 subtractor **28** would not operate because a carry is generated in the addition from position 4 to position 5 and gate **26** is inhibited from operating. Consequently the digit which remains in the four

least significant stages of the adder is 0001, the BCD equivalent of the decimal digit 1.

The circuit of FIG. 1 is provided with circuitry for detecting carries from position 4 to position 5 and from position 8 to position 9 in the adder. Detectors **34** and **35** are not even necessary in the circuit because, as described above, if carry detectors are provided, their operations alone determine which digits need correction; if a carry is not generated the digit has 0110 subtracted from it. Thus in the circuit of FIG. 1 detectors **34** anad **35** may be omitted and gates **26** anad **27** may have only the one input connected to controller **20**. Many data processors however are provided with binary adders in which the carries generated are internal and cannot be detected. The circuit of FIG. 3 is another illustrative embodiment of the invention in which the sum derived by the adder may be corrected without requiring the detection of carries. The basic steps carried out in the circuit of FIG. 3 are similar to those carried out in the circuit of FIG. 1. There are a few major differences however between the two circuits. The organization of the data stored in memory **25** is that shown in FIG. 4. Binary adder **23** now comprises 20 stages even though the final BCD result appears in only the **12** least significant stages. Gate **27** is no longer provided with an inhibit terminal. Finally, subtractor **28** is controlled not only by gate **26** (which is also no longer provided with an inhibit terminal), but in addition by gates **30**, **31** and **32** and detector **33**.

The organization of the data in the memory of FIG. 3, as shown in FIG. 4, is similar to the organization of data shown in FIG. 2. However the memory no longer comprises two 32-word tables, one having eight-bit words and the other having 12-bit words, but instead comprises one 32-word table where each word has 20 bits. Each word in the table represents five BCD digits. The two most significant digits are the same as the two digits in the right hand table of FIG. 2 for the respective memory location. The three least significant BCD digits are the same as the digits stored in the left hand table of FIG. 2 at the respective address. When addresser **24** transmits the five least significant bits in the input binary word to the memory the memory transmits the eight most significant bits in the respective location to the eight least significant stages in the adder. Since the eight most significant bits in each location in the table of FIG. 4 are the same as the eight bits at the respective address in the right hand table of FIG. 2, it is seen that the five least significant bits in the input binary word control the transmission to the adder of the same two BCD digits in the two circuits. When the addresser transmits the five most significant bits in the input word to the memory the memory transmits the entire 20-bit word in the respective memory location to the 20 stages of the adder. The 12 least significant bits transmitted to the adder in step (10) are the same as those transmitted to the adder in step (10) in the circuit of FIG. 1. The major difference between the data transmitted to the adders in the circuits of FIGS. 1 and 3 is that in the latter the BCD excess-3 equivalent of the five least significant bits in the input word is transmitted not only to the eight least significant stages in the adder but also to the eight most significant stages in the adder. Even though the bits stored in these latter eight stages are not part of the final BCD result one of these bits is used in place of a carry detector mechanism of FIG. 1.

The sum which appears in the first 12 stages in the adder of FIG. 3 is the same as that which appears in the adder of FIG. 1 after the addition operation. It will be recalled that the BCD digit in stages **9** through **12** need not be corrected. However the number 0110 may have to be subtracted from the digits appearing in stages **1** through **4** and **5** through **8**. The governing rule is the same as that described above: The number 0110 is subtracted from each of the first two digits if the digit is equal to or greater than 0110 and a carry was not generated in the addition step from the respective position to the next higher position.

The correction mechanism for the second digit in the adder of FIG. 3 is the same as that in FIG. 1 except that no inhibit terminal is provided for gate **27**. The purpose of the inhibit terminal is to prevent subtractor **29** from operating even though the second digit is 6 or greater if a carry was generated to the third BCD digit position in the adder. The reason that the inhibit terminal, i.e., carry detection, is not required is that if a carry was generated to the third BCD digit during the addition, the second BCD digit in the adder must be less than 6. Consequently there is no reason to detect the carry since detector **35** prevents the operation of AND gate **27** and subtractor **29** whenever a carry is generated. It will be recalled that the initial binary number to be converted is separated into two parts, $32A+B$. B is a two digit BCD number lying within the range 0 through 31. The "tens" digit, the digit which is transmitted to stages **5** through **9** in the adder, is thus 0, 1, 2 or 3. The digit appears in the excess-3 code in the table of FIG. 4 and thus the digit which is actually transmitted to the second BCD position in the adder is 3, 4, 5 or 6. The "tens" digit in the 32A factor falls within the range 0 through 9, or 3 through 12 in excess-3. Referring to FIG. 4 it is seen that the column containing the fifth through ninth bits in the table contain the BCD excess-3 numbers 3 through 12. The addition of one of the numbers 3–6 and one of the numbers 3–12 produces a sum which lies in the range 6 through 18. Since a carry may be generated from the fourth position in the adder the actual sum derived in the second BCD digit position in the adder is in the range 6 through 19. The four bits in this position, however, can represent a maximum number of 15 and thus if the derived sum is one of the numbers 16 through 19 a carry will be generated to the third BCD position and the second position will represent one of the numbers 0 through 3. Thus the second BCD digit is one of the numbers 6 through 15 if no carry is generated and is one of the numbers 0 through 3 if a carry is generated. Subtractor **29** must operate only if the second BCD digit in the adder is equal to or greater than 6 and a carry was not generated. Since the digit is equal to or greater than 6 only when a carry is not generated there is no necessity for detecting the carry. If the digit is equal to or greater than 6 the number 6 must be subtracted from the second digit. For this reason it is not necessary to detect the generation of a carry from position 8 to position 9 and the inhibit terminal of gate **27** may be omitted.

The correction of the "'units" digits in the adder is more complicated. The "units" digit of the number B transmitted to the adder lies in the range 0 through 9 in BCD, and 3 through 12 in the excess-3 code. The "units" digid in the number 32A which is transmitted to the adder is one of the numbers 0, 2, 4, 6 and 8, or one of the numbers 3, 5, 7, 9 and 11 in the excess-3 code. The range of the "excess-6" sum derived in the first four bit positions of the adder is thus 6 through 23, or 6 through 15 and 0 through 7, the latter range being associated with the generation of a carry bit to the fifth bit position in the adder.

Since there are two distinct ranges—6 through 15 which is not associated with the generation of a carry bit and 0 through 7 which is associated with the generation of a carry bit—ambiguity exists only for digits 6 and 7 where the two ranges overlap. If the digit in the adder is 8 or greater no carry was generated. Consequently if the digit is 8 or greater the number 6 should be subtracted from the least significant BCD digit in the adder. If the digit is 5 or less the number 6 should not be subtracted from it since this number is never subtracted from a digit which is less than 6 in value. However if the digit in the adder is 6 or 7 it is not self-evident whether the number 6 should be subtracted from it. A 6 may result in one of two ways—by the addition of 3+3 and by the addi-

tion of 11+11, the latter generating a carry. In the first case subtractor **28** should operate since no carry was generated; in the latter case it should not. Similarly, the number 7 can be produced by adding 3+4 and 11+12, the latter generating a carry. In the former case subtractor **28** should operate; in the latter case it should not. Thus if the first BCD digit in the adder is 6 or 7 it is not possible to determinne whether the number 6 should be subtracted from it merely by examining the digit itself. If a carry is generated the result should be left unchanged. If a carry was not generated the sum derived should be corrected. Since the circuit of FIG. 3 does not include a carry detector connected between positions 4 and 5 of the adder it is necessary to determine whether a carry was generated in some other fashion.

As described immediately above the sum digit 6 results from the addition of 3+3 or 11+11. The sum digit 7 results from the addition of 3+4 or 11+12. A carry is generated only in the second case of each pair. In both cases where a carry is generated the digit 1011 is transmitted to the "units" position of the adder as part of the 20-bit number 32A. In both cases where a carry is not generated the digit 0011 is transmitted to the "units" position of the adder as part of the 20-bit number 32A. It is possible to examine the four least significant bits in the 32A word to determine if they represent decimal 3 or decimal 11, i.e., to determine if the "units" digit should be corrected. However, there is an easier way to determine if the "units" digit needs correction. An examination of the table of FIG. 4 discloses the following: Whenever the four least significant bits in one of the 20-bit data words represent the decimal number 11 the fifteenth bit position of the same word is a 1, and whenever the four least significant bits represent a decimal 3 the fifteenth bit in the word is a 0. Consequently of the final digit in the "'units" position of the adder is a 6 or a 7 a carry was generated from position 4 to position 5 in the addition only if the fifteenth bit in the adder is a 1. In other words, subtractor **28**, which should operate only if a carry was not generated, should operate only if the fifteenth bit is a 0. In the circuit of FIG. 3 if the resulting "units" digit is a 6 or 7 subtractor **28** operates only if the fifteenth bit in the adder sum is a 0. (It should be noted that the fifteenth bit in the adder will always be the same as the fifteenth bit in the 32A number transmitted to the adder which contains decimal 11 in the least significant four bits because no carries may be generated from position 12 of the adder to position 13; the sum, or 32A+B, is always equal to or less than decimal 999 and consequently a carry is not generated to position 13 of the adder which could affect the bit in position 15.)

As described above if the "units" digit is equal to or greater than 8 subtractor **28** should operate. It is seen that in such a cast detector **24** operates and in step (11) gate **26** operates to transmit an output pulse through OR gate **32** to enable subtractor **28**. If the "units" digit in the adder is 5 or less subtractor **28** should not operate and as seen in FIG. 3 since neither detector **24** nor **33** operates no pulse is transmitted through OR gate **32** to initiate the operation of the subtractor. If the "units" digit is 6 or 7 detector **33** operates and enables one input of AND gate **31**. In step (11) of the machine operation gate **30** operates to enable the other input to gate **31** only if the fifteenth bit in the adder is a 0. When gate **31** operates a pulse is transmitted through OR gate **32** to operate subtractor **28**. Thus it is seen that in the embodiment of FIG. 3 it is not necessary to detect carries to determine if a digit needs correction. Because more bits are transmitted to the adder than are actually required in the BCD result one of these bits may be examined to determine if a carry was actually generated during the addition. The same technique may be applied in other applications where the input binary number to be converted may have a value greater than decimal 999. In all cases the data table may be constructed and two pieces of data may be transmitted to the adder in such a manner that an examination of the sum derived by the adder may in itself determine whether carries were generated from the digit positions of concern.

The circuits of FIGS. 1 and 3 are designed to accomplish a single result, namely, the conversion of a 10-bit binary number having one of the decimal values 000 through 999 to its 12-bit BCD equivalent. The method of the invention however can also be practiced to great advantage on general-purpose data processors. The block diagram for such a machine is shown in FIG. 5. A data processor along the lines of that shown in FIG. 5 may be constructed from standard units well known in the art. For an understanding of how the method of my invention may be practiced on such a machine it is necessary to describe only briefly the mode of operation of the system of FIG. 5, and certain of the orders which it may execute.

The system includes a program store **40** which contains instruction words and a data store **41** which contains data words. Instruction words and data words are transmitted back and forth between these two units and control **42**. The control governs the system operation in accordance with the instruction words transmitted to it and in accordance with the values of particular data words which it retrieves from the data store. The control can govern the data store to transmit a data word through the masking and exclusive-OR circuits **43** to register selector **44**. The register selector in turn directs the output data word from the masking and exclusive-OR circuits to one of registers A, B and C. To perform an exclusive-OR or masking operation two data words are required. One data word is transmitted to circuits **43** from data store **41**. The other is transmitted to circuits **43** from one of registers A, B and C by register reader **45**. The register reader also directs a register word to data store **41** if the data word is to be written in the store. If two data words are to be added together they are placed in two of registers A, B and C and added by adder **46**. The sum is directed to register director **44** which in turn directs the sum to one of the registers. A register word may also be shifted by shift circuit **47** under command of control **42**.

The exclusive-OR operation is performed on two data words. Corresponding bits are compared and a respective bit in the output word is a 1 if only one of the two corresponding bits in the two data words being operated upon is a 1. In the masking operation the output bit is a 1 only if both of the corresponding bits in the two data words being operated upon are 1's. Suppose for example it is desired to substitute the fifteenth bit in one word for the first bit of a second word. The second word may be masked by a mask word which contains all 1's except for the least significant bit which is a 0. After passing through circuits **43** and stored in a register the second word will be unchanged except for the first bit which will be a 0 whether or not the first bit was initially a 0 or a 1. The first word may then be stored in one of the machine registers and shifted 14 positions to the right so that the fifteenth bit in the original word is now the least significant bit in the register. The resulting shifted word may then masked by a mask which contains all 0's except for the first bit which is a 1. If the resulting masked word is stored in a register the register word will contain all 0's except for the first bit which is the same as the fifteenth bit in the original word. If this word is now added to the second word which is contained in one of the registers (with its first bit a 0) the sum will be the initial second word with the first bit having been substituted by the fifteenth bit of the first word.

The method of the invention may be practiced on the data processor of FIG. 5 if data store **41** contains two tables, the first being that shown in FIG. 4 and the second being that shown in FIG. 6. Each of the data words in FIG. 6 is eight bits in length, with the decimal equivalents

of each pair of four-bit sub-words being shown in parentheses. The 32 words in the table of FIG. 6 are in successive locations beginning with the first at address $N+0000$. The method described above is carried out on the machine of FIG. 5 up to the point where adder **46** derives the sum of $32A+B$. It should be noted that when the word B is transmitted from data store **41** to one of the registers the 12 least significant bits are masked out so that only the eight most significant bits are stored in one of the registers. These eight bits are then shifted to the right to the eight least significant stages of the register. When the number 32A is transmitted from data store **41** it is not masked and the full 20-bit word appears in one of the registers. The first 12 bits in the sum word derived by the adder and directed by register selector **44** to one of the registers comprise the final BCD result except that the number 0110 should be subtracted from the second group of four bits if their value is 6 or greater, and the number 0110 should be subtracted from the first group of four bits if the value of these bits is 8 or greater, or if the value of these bits is 6 or 7 and the fifteenth bit in the sum word is a 0.

The sum word may be operated upon to correct the second group of four bits in a manner well known in the art. The sum word may be shifted four positions to the right and original bits 9 through 20 masked out. The resulting word contains all 0's except for the first four bits which represent the "tens" digit in the final result, in the BCD excess-6 code. Control **42** then determines if this digit is equal to or greater than 6 and if it is subtracts the number 0110 from it. (This can be accomplished by adder **46** by the use of 2's complement arithmetic.) The final result then may be shifted four positions to the left and through a masking operation be substituted for the four bits in the initial sum word which is retained in another register.

The "tens" digit in the sum word may also be corrected as follows: The four bits comprising this digit are examined. The four bits are added to the number N and the four leftmost of the eight bits at the respective location in the table of FIG. 6 are retrieved. These four bits are exclusive-ORed with the four initial bits and the result is the required "tens" BCD digit. For example, suppose the initial BCD digit in the sum word is 1100. The four leftmost bits in the table of FIG. 6 at location $N+1100$ are 1010. When the numbers 1100 and 1010 are exclusive-ORed together, the result is 0110. It will be noted that the initial BCD digit 1100 is decimal 12. Since this number is greater than 6, the number 6 should be subtracted from it and the final BCD digit should be 0110. This is in fact the result obtained after the exclusive-OR operation. The table of FIG. 6 is constructed in such a manner that the exclusive-OR operation in effect controls 0110 to be subtracted from the initial digit if necessary, i.e., if the digit is 6 or greater.

After the second BCD digit in the sum is corrected by either of the two procedures described above, the first digit is corrected. A highly efficient correction sequence for this sum digit is the following: The fifteenth bit in the data store word containing the original 32A word is retrieved from the data store and this bit is substituted for the least significant bit in the sum digit. The four least significant bits in the modified sum word are then added to the number N. The resulting number represents an address in the table of FIG. 6 and the four least significant bits at the respective location are exclusive-ORed with the four least significant bits in the original BCD sum word. The result is the required "units" BCD digit. For example, suppose the four least significant bits in the sum word are 1100. Since this number is greater than 8, as described above, a carry cannot be generated during the addition of 32A+B. When either a 1 of a 0 is substituted for the least significant bit in the four-bit word 1100, the result is 1101 or 1100, respectively. The four least significant bits stored at address N+1100 and N+1101 in the table of FIG. 6 are both 1010. When the numbers 1100 and 1010 are exclusive-ORed together the result is 0110, decimal 6, the required "units" BCD digit since the initial digit was decimal 12. Suppose, on the other hand, the initial "units" BCD digit was 1101, decimal 13. Again, there was no carry, and regardless of the value of the fifteenth bit in the result, the number 1010 is retrieved from the table of FIG. 6 and exclusive-ORed with the initial digit 1101. The result 0111, decimal 7, is the correct value.

Consider next a case where the initial derived "units" BCD digit is 5 or less. Here, as described above, a carry must have been generated. Regardless of the value of the fifteenth bit that is substituted for the least significant bit in any one of the numbers 0000 through 0101, the result, when added to N, identifies in the table of FIG. 6 one of the first five addresses, all of which contain 0000 in the four least significant bit positions. When 0000 and the initial BCD digit are exclusive-ORed together, the final digit is the same as the initial one as required since the initial digit, 5 or less, should remain unchanged.

The last case to consider is the one in which the "units" digit appearing in the adder is 6 or 7. If it is a 6 the four least significant bits are 0110. The fifteenth bit can be either a 1 or a 0. If it is a 0, it serves as an indication that no carry was generated and when a 0 is substituted for the least significant bit in the digit the digit is still 0110. The four least significant bits in the eight-bit number stored at address N+0110 in the table of FIG. 6 are 0110. When the initial digit 0110 is exclusive-ORed with the table entry 0110 the result is 0000. Since a carry was not generated the value 6 should be subtracted from the initial digit to produce the final digit 0. On the other hand, if the fifteenth bit is a 1, it serves as an indication that a carry was generated and when a 1 is substituted for the least significant bit in the initial digit the result is 0111. The table entry used for this value, i.e., the group of four least significant bits at address N+0111 in the table, is 0000. When the initial digit 0110 is exclusive-ORed with 0000 the result is 0110. This is the desired result because the initial digit should remain unchanged since a carry was generated from position 4 to position 5 in the adder.

If the initial digit is a 7, a binary 0111, and the fifteenth bit is a 0, the entry at address N+0110 is retrieved. This number, 0110, is exclusive-ORed with the initial digit 0111. The result 0001 is 6 less than the initial digit. This is the required BCD digit since a carry was not generated and the value 6 should be subtracted from the "units" digit in the adder. Finally, if the fifteenth bit is a 1 the table entry at address N+0111 which is used in the exclusive-OR operation is 0000. When 0111 and 0000 are exclusive-ORed together the result is 0111. The initial digit remains unchanged since a carry was generated.

It should be noted that many alternative tables can be used instead of that shown in FIG. 6 if operations other than the exclusive-OR operation are performed to modify the "tens" and "units" digits. On the other hand, it is not necessary to use table entries at all in correcting the "tens" and "units" digits if arithmetic operations are carried out following the basic rule: The number 0110 is subtracted from the "tens" digit if the digit is equal to or greater than 6; and the number 0110 is subtracted from the "units" digits if it is 8 or greater, or if it is 6 or 7 and the fifteenth bit in the sum word is a 0.

The method of the invention is also applicable to other code conversions. For example, suppose it is desired to convert the initial binary number to the respective 15-bit 2-out-of-5 code. In the 2-out-of-5 code a 5-bit number represents a decimal digit. Each 5-bit value has two 1's and three 0's. The weights of the digits in the 2-out-of-5 code from left to right are 7, 4, 2, 1 and 0. In forming the 2-out-of-5 equivalent of a decimal digit 1's are placed in the two positions having the weights whose sum is equal to the decimal digit. For example, the digit 9 is

represented by the 2-out-of-5 code 10100 and the digit 4 is represented by 01001. The only exception is the digit 0 which is represented by 11000.

It is not possible to modify the 12 least significant bits in the adder to derive the 15-bit 2-out-of-5 equivalent of the binary number to be converted because there is no correspondence between the 12 bits in the adder and the 15 bits required in the final result. For these reasons each of the three 2-out-of-5 digits is constructed individually by the use of a table such as that in FIG. 7. The table includes 32 entries stored in successive locations beginning at address N+0000. Each entry comprises two 2-out-of-5 numbers. Adjacent each entry in brackets are the decimal equivalents of each entry.

The steps of my method are carried out through the derivation of the BCD sum in the adder. Then the "hundreds" digit is constructed. Digits 9 through 12 in the adder represent the BCD equivalent of the required "hundreds" digit. The number N+0110 is added to this 4-bit number and the resulting number is an address in the table of FIG. 7. The five rightmost bits at the respective location represent the required 2-out-of-5 "hundreds" digit. For example, suppose the BCD digit in the adder is 0010, decimal 2. When the number N+0110 is added to this value the resulting address is N+1000. The five least significant bits at this address are 00101, the 2-out-of-5 code for decimal 2.

The "tens" digit is derived by adding the number N to the second BCD digit in the adder to derive an address. The respective five rightmost bits in the table of FIG. 7 represent the required 2-out-of-5 digit. For example, if the BCD digit in the adder is 0010 the entry retrieved from tht table is 00101. It should be noted that the rightmost 2-out-of-5 digits in the table increase from 0 through 5 and then start once again with 0 and end at 9. If the "tens" BCD digit is 5 or less, one of the first six entries is retrieved, the entry being the 2-out-of-5 equivalent of the BCD digit. On the other hand, if the BCD digit is 6 or greater the number 6 should be subtracted from it to derive the correct digit value. Each of the right hand table entries beginning at address $N+0110$ already has the value 6 subtracted from the respective address and thus represents the required 2-out-of-5 digit code.

The "units" digit is constructed as follows. The number N is added to the "units" BCD digit in the adder to derive an address in the table of FIG. 7. If the "units" digit is 5 or less, or 8 or greater, the "units" digit is treated just as the "tens" digit. For this reason if the "units" digit is 5 or less or 8 or greater the right hand column (or left hand column since they are equal for addresses equal to $N+0101$ or less and $N+1000$ or greater) in the table of FIG. 7 is used. The respective entry is the corrected 2-out-of-5 equivalent of the "units" BCD digit in the adder. If the BCD digit in the adder is a 6 or 7 a choice is made between the left hand and right hand entries at locations $N+0110$ and $N+0111$. If the BCD digit in the adder is a 6 or 7 and the fifteenth bit is a 0 the two respective righthand entries 11000 and 00011 (0 and 1 respective in the 2-out-of-5 code) are used to construct the 2-out-of-5 digit. If on the other hand the fifteenth bit is a 1 the two respective left hand entires 01100 and 10001, 6 and 7 in the 2-out-of-5 code, are used to construct the "units" digit.

Thus it is to be understood that although the invention has been described with reference to certain particular embodiments numerous modifications may be made therein and other arrangements may be devised without departing from the spirit and scope of the invention.

What is claimed is:

1. An arrangement for converting 10-bit binary numbers to their binary coded decimal equivalents in a data processor including a binary adder with at least 15 stages and means for storing in 20-bit memory locations in said processor a series of 32 numbers, the eight most significant bits in each number of said series being the excess-3 binary coded decimal equivalent of a respective one of the binary numbers 00000 through 11111 and the 12 least significant bits in each number of said series being the binary coded decimal equivalent of the same respective 5-bit binary number multiplied by the binary number 100000 with the eight least significant bits in each number of said series also being in the excess-3 code, said arrangement comprising

means for registeritng said 10-bit binary number to be converted,

means for transmitting to the eight least significant bit positions of said binary adder only the eight most significant bits of the memory number in said series corresponding to the five least significant bits of said registered 10-bit binary number to be converted,

means for transmitting to said binary adder the complete 20-bit memory number in said series corresponding to the five most significant bits of said registered 10-bit binary number to be converted, said binary adder thereupon adding said numbers transmitted thereto to form a sum,

means for subtracting the binary number 0110 from the second four least significant bits in said sum when the number represented by these bits is 0110 or greater,

means for subtracting the binary number 0110 from the four least significant bits in said sum when the number represented by these bits is equal to or greater than 1000, and when the number represented by these bits is 0110 or 0111 and the 15th bit in said sum is a 0, and

means for extracting from said sum the 12 least significant bits to represent the binary coded decimal equivalent of said 10-bit binary number to be converted.

2. An arrangement for converting r-bit binary numbers each composed of an n-bit part and an m-bit part into binary coded decimal numbers comprising a data processor having a binary adder and a memory unit, said memory unit having stored therein successive locations a plurality of binary coded decimal equivalent words, the first portion of each of said locations having contained therein the binary coded decimal equivalents of one of the binary numbers 0 through 2^n-1 and the second portion of each of said locations having contained therein the binary coded decimal equivalent of one of the binary numbers $2^n(0)$ through $2^n(2^m-1)$, at least some of said binary coded decimal equivalent numbers being stored in the excess-3 code,

means for transmitting to the least significant bit positions of said binary adder the contents of only the first portion of one of said locations corresponding to said n-bit part of said number to be converted,

means for transmitting to said adder the contents of the first and second portions of one of said locations corresponding to said m-bit part of said binary number to be converted, said adder thereupon adding said numbers transmitted thereto to form a sum,

means for ascertaining the value of one bit in said sum lying in a bit position outside the range of bits representing the converted binary coded decimal number, and

means controlled by said ascertaining means for selectively subtracting 0110 from the digits in said sum.

3. A data processor for converting binary numbers to their binary coded decimal equivalents comprising

a data processor memory unit having stored therein a table containing the binary coded decimal equivalents of a series of binary numbers, said series of binary numbers being such that each of the binary numbers to be coverted is equal to the sum of two of the binary numbers in said series, predetermined ones of said numbers in said table being a fixed multiple of others of said numbers in said table and at least

some of said numbers in said table being stored in the excess-3 code,

binary adder means for adding together the two binary coded decimal numbers in said table which are the equivalents of the two binary numbers whose sum is equal to the binary number to be converted, and

means for adjusting at least one of the binary coded decimal digits in the sum derived by said binary adder means in accordance with the value of said one digit and the value of at least one other predetermined bit in said sum.

References Cited

UNITED STATES PATENTS

3,189,735	6/1965	Gunderson et al.	235—169
3,271,566	9/1966	Martens	235—174

MAYNARD R. WILBUR, Primary Examiner

G. R. EDWARDS, Assistant Examiner

U.S. Cl. X.R.

235—157; 340—347