COMPANY CONFIDENTIAL

7000 SERIES
FILE MEMO NO. 62

June 3, 1957

Subject: Fingers or Fists?

(The Choice of Decimal or Binary Representation for the 7000 Series)

By: W. Buchholz

Introduction

Much thought has been given by many individuals to the choice of decimal or binary representation in 7000 Series equipment. There exists general agreement that the basic representation should be binary, but that a decimal mode of arithmetic should also be provided. The subject is not a simple one. A number of inter-related considerations combine to lead to this conclusion. These include the representation of numeric data, non-numeric data and instructions, variable field and record length, manipulation of individual bits, and transformation by table look-up.

1. Numeric Data

Numeric data for business data processing are almost exclusively in decimal form. Only few arithmetic operations are performed on each number. Decimal arithmetic avoids a great deal of decimal-binary conversion.

For technical computation, decimal floating-point numbers are entirely usable and have the advantage of ease of human interpretation. Binary floating-point numbers, however, show a clear advantage in greater performance for a given information content and equipment complexity:

1.1 Greater arithmetic speeds.

1.2 Less round-off error per operation.

1.3 More efficient memory utilization.

1.4 Greater external storage (tape or disk) speed for a given bit transmission rate.

2. Binary Information

A great deal of information in commercial usage is non-decimal, and computers adept at handling binary information - the common denominator - will be much superior to similar machines with a more limited repertoire. A list of non-decimal kinds of information includes:

2.1 Alphanumeric codes used internally and in other systems.

2.2 Paper tape codes.

2.3 Data transmission codes.

2.4 Card punching other than "standard" (controls, MLP codes, signs, multiple-hole codes, "underpunching", etc.).

2.5 Output of character sensing scanners and similar devices.

2.6 Instructions (other than the addresses which may be decimal or binary).

2.7 Data for logical and decision operations.

2.8 Machine status information.

2.9 Switches and lights under program control.

2.10 Analog-digital conversion and telemetering.

3. Character Codes

6-Bit character codes have been used throughout the 700 Series because the number of different characters to be represented was less than 64 and because uniformity is very desirable.

In reviewing the choice of character code, one notes that even the 700 Series codes are not completely uniform. Other manufacturers have adopted quite different codes, even those who use a 6-bit format. There are actually some serious disadvantages to adopting a uniform 6-bit code for all alphabetic and decimal characters.

3.1 A 6-bit code is inefficient for purely numeric decimal data. For many large files in commercial usage, a 4-bit decimal code provides a substantial memory saving and increase in performance.

3.2 64 Different characters are not necessarily sufficient. We should look ahead to extended use of symbols, lower-case letters, etc. While this is not a pressing need at this time, we may be sure that expanded alphabets will be used as soon as present equipment constraints are relaxed. After that, it will not be long before the feature becomes indispensable.

3.3 For data transmission, we may expect 4-out-of-8 codes to be used. A machine capable of handling 8-bit characters can directly translate this code without extra hardware.

3.4 The reason why the prevalent 705-727 6-bit code is not now completely standard is that it does not permit all 64 combinations. Moreover, the desired collating sequence is awkward to achieve. It seems unlikely that the computer industry would ever agree to the present IBM code as standard. IBM has an obligation to clean up its own code situation to facilitate future standardization.

The most desirable solution from the user's point of view is to permit a variable character code, both as to the number of bits and their meaning. To reduce hardware, we may wish to compromise in a basic commercial system and adopt one of the following schemes.

3.5 Permit 4-bit decimal digits and 8-bit (2 decimal digit) alphanumeric characters.

3.6 Permit 4, 6, or 8 bit codes.

3.5 is the simpler of the two schemes to mechanize and it is compatible with binary addressing. 3.6 is more efficient when alphabetic data predominate. It should be noted that 3.5 is probably no less efficient overall than a uniform 6-bit code.

4. Variable Field and Record Length

The concept of variable field and record length, established in the 702 and 705, has proved of great value to business data processing and should be considered as fundamental to any future system.

The feature has not, so far, been available for high-speed scientific computing because it did not appear to offer direct benefits in floating-point operations. When considering problems too large to fit into the working memory, however, it becomes possible to obtain greater storage efficiency and speed of information transfers by taking advantage of variable field and record length for storing data on tapes or disks. Other benefits arise in manipulating parts of data and instruction words.

To avoid programming complexity in using the variable length feature, the smallest accessible unit of information must be directly addressable. (In the 702 and 705 this unit is the 6-bit character.)

Regardless of the choice of unit, the addresses should span the entire memory. From the programmer's point of view, there should be no break apparent at the boundaries of the word size chosen for memory. The memory word boundaries are artificial and exist only for greater efficiency in memory design. From the designer's point of view, the address structure should be chosen so that address bits, in some suitable code, can be easily split into a memory word address and an address within the word. (In the 702-705, the 5-character core-memory word size agrees with the decimal address structure.)

5. Bit Addressing

An extension of the variable field length concept is to permit direct addressing of a single bit, or a group of any number of bits, for logical manipulation. Signs, control punches in cards, and program switches are examples where direct bit addressing is beneficial. The principle extends to evaluating logical expressions consisting of a sequence of AND, OR, and similar functions of binary variables. This technique is intended to replace the cumbersome "Christmas trees" of conditional instructions which are responsible for much of the bulk and complexity of present programs.

Another important application of bit addressing is in tables of single-bit, yes-no indicators. Practical examples occur in check reconciliation and compressing storage of large arrays of data containing mostly zero elements.

An alternative to bit addressing is the masking technique used in the 704 and 709, but this is usually an indirect and inefficient technique. The saving in address bits possible by eliminating bit addresses is not enough to offset the disadvantages of alternative schemes.

The reasons leading to bit addressing also lead to the need for indexing bit addresses.

6. Transformation

Von Neumann introduced, about 1945, the basic concept of the stored program where instructions, and particularly addresses, are stored and processed by the same means as the data. A most important new concept has been added more recently, that of transforming data into addresses for the purpose of table look-up. (This concept was recognized as a basic computer operation by Amdahl, Boehm, and Griffith during the early definition of the 709 in 1955, and it appears in the 709 in the form of

"Convert" instructions. The first written reference appears in Stretch Memo No. 16. Limited application of the scheme to programmed decimal-binary conversion dates back to the 701 project in 1951.)

The transformation type of table look-up, as distinguished from the table searching form used in the 650, is the most general and efficient method of code conversion available. It has broad applications in editing, particularly when the rules of conversion are subject to change and exceptions. Built-in provisions for table look-up, including a path from the data section to the address section of the computer, will permit substantial economies elsewhere by removing the need to build editing and code conversion into external equipment.

Since the data to be transformed are mostly non-decimal, the transformation operation must be able to deal with binary data. This fact virtually dictates binary addressing. It is true that decimal addressing does not entirely preclude transformation of binary data by indirect methods, but they are very wasteful of time or memory space.

7. Addressing Requirements

Sections 3 to 6, above, contain the major factors entering into the choice of the addressing system for the 7000 Series.

7.1 Character code considerations lead to 4 bits as the smallest accessible code unit (called byte) for representing decimal information. 8-Bit, and possibly 6-bit, units or bytes are also needed.

7.2 Bit addressing leads to 1 bit bytes as the smallest addressable information unit. By choosing 1 bit as the smallest addressable unit, it becomes also possible to address 6-bit character codes as directly as 4 and 8-bit codes.

7.3 Variable field and record length requires that a large enough block of memory be addressable continuously, in steps of the smallest addressable unit, here 1 bit. The addresses must cross memory word boundaries without gaps.

7.4 Data transformation for table look-up requires that the addresses be binary numbers. Binary addressing also has the advantage, over comparable decimal addressing, of greater compactness and economy.

7.5 Design considerations require that the memory word size (excluding check bits, etc.) be a simple multiple of the smallest addressable unit, here a power of 2. Practical reasons limit the choice to 32 or 64 bit words, and the 64-bit word was selected as a compromise between cost and performance.

7.6 Hardware economy also dictates a standardization of byte size outside of the computer. While some high-performance external units will communicate with memory one full 64-bit word at a time, most units can be handled more economically, via a common Exchange, with a much smaller byte size. To keep the common Exchange as simple as possible, the standard byte size must also be a power of 2, and an 8-bit byte was chosen. The 8-bit byte readily accommodates the common 6-bit character code by the insertion of two zero bits.

8. Program Debugging

Decimal addressing has the advantage of simplifying visual interpretation when program debugging is done on the machine. This advantage is heavily diluted by the difficulty of interpreting the non-decimal remainder of the instruction as well as non-decimal data. This difficulty exists regardless of the choice of address radix. Decimal addressing does not obviate the need for standard programs to assist in debugging, and these programs can be readily enlarged to include binary-to-decimal (or octal) conversion when addresses are binary.

9. Competitive Considerations

It is realized that there is a considerable feeling in favor of decimal machines. What is commonly overlooked, of course, is that the only true decimal machines ever built are purely numeric machines. The 705, the Univac, and similar machines obey special and decidedly non-decimal rules for alphanumeric characters. The more powerful the machine, the more do binary features creep into the design, as witness the multiple use of zones in the 705. And even the decimal 604 has a strictly binary plugboard on the punch, which is really what permits the 604 to deal with complex punching in cards.

We must, of course, expect a competitor who has a decimal machine to try to exploit a common prejudice. On the other hand, a superior machine is much more formidable competition. A machine with binary addressing can clearly be superior in performance, versatility, and

simplicity to a decimally addressed machine using similar components. We cannot leave ourselves open to that kind of a competition.

Conclusions

Binary addressing has a clear advantage over decimal addressing in that it combines greater performance with less complex hardware. Contrary to a popular opinion, binary addressing appears to be of greater benefit to business applications than to scientific computing where it is mostly a matter of cost.

For business applications, binary addressing permits transformations of non-decimal data directly and efficiently. Such facilities have never been available to commercial customers, but they are expected to gain a great deal of importance in connection with routine re-arrangement, conversion, and editing of data by stored program.

Standard word and byte sizes which are a power of 2 follow as a consequence of choosing binary addressing. The actual choices are 64 and 8 bits, respectively.

Numeric data in business applications are normally in decimal form, and a decimal adder is highly desirable even with a basically binary machine. Binary arithmetic is also needed to operate on addresses.

For scientific applications, binary numeric representation provides greater performance and more efficient use of storage.

Thus, the 7000 Series will be binary throughout except for a second decimal adder in the serial computer.