

April 5, 1957

TO: Mr. S. W. Dunwell

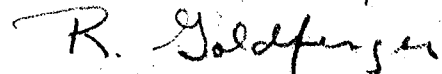
SUBJECT: Choice of Byte Size in the Commercial Harvest Machine

Attached is a memorandum which is indicative of our current thinking on the subject of byte size for a commercial machine. It is in response to your bringing up the question on March 21.

As we have been careful to state in the memorandum such conclusions as have been reached are predicated upon the assumptions of a 64-bit word and binary addressing, as well as other constraints now planned for the Harvest system. Were the assumptions to change, our conclusions might change also.



E. F. Codd



R. Goldfinger

EFC:RG:jv

Attachment

April 5, 1957

File Memorandum - STRETCH

Subject: Choice of Byte Size in the Commercial Harvest Machine

(Investigation of this subject was requested by Mr. S. W. Dunwell on 3/21/57)
The scope of the investigation was necessarily limited by lack of information on the Harvest Machine.

1. INTRODUCTION

In order to reduce the cost of the commercial model of the Harvest Machine, the byte, instead of being program-variable in length from one to eight bits, will be limited to a few standard lengths. While this investigation assumed an absence of preconception as to the optimum byte size choice, attention was soon focused on the 1-4-8 representation. That is, the elementary unit of information called "byte" may comprise one bit (binary), four bits (decimal numeric), or eight bits (alphabetic, special characters, decimal). It was felt that unless serious objections could be raised to the 1-4-8 choice, and in the investigation such objections were sought, the 1-4-8 choice represented the optimum in economy in treating strictly numerical information, and the optimum in capacity for a variety of alphabets, special characters and operational symbols. This report, then, is concerned principally with justifying the choice of a 1-4-8 byte size range and introducing some of the problems anticipated in the handling of bytes in these three sizes in the commercial Harvest System.

2. ASSUMPTIONS

These assumptions about the commercial Harvest System are made for the purpose of limiting the discussion to one possible overall machine system. Any argument contained in, or deriving from this report, can be invalidated or demolished merely by altering the reference frame in which it has been so tenuously hung. Therefore, to provide a single, typical system for discussion the following hypothesized commercial Harvest system is presented:

- word length - 64 bits
- addressing - binary
- index and control quantities - binary
- data - binary, decimal, alphabetic, special
- field length - variable, extensible over word boundaries
- byte - not extensible over word boundaries
- arithmetic units - binary and decimal
- logical abilities - binary, decimal, alphabetic
- exchange - communicating with memory in eight bit bytes providing 6 to 8 or 8 to 6 bit byte conversion in going to or coming from I/O units where required

3. THE PROBLEM

Which limited set of byte sizes shall be adopted for the commercial Harvest System? From the assumptions stated it is apparent that the final set must include the one-bit byte for binary representation. There are three other possible and reasonable choices for the representation of decimal and alphabetic information. A minimum of four bits are required for decimal, although decimal may utilize as well six or eight bits, penalized only by a loss of storage efficiency.

Alphabetic information by which is meant any number of alphabets, cases, special characters, programming symbols, etc., may be represented by six or eight bits. The eight bit byte is our first choice in this case. It excels in capacity; it is compatible with present exchange specifications. The number of bits per byte is a power of two and is therefore, a divisor of the word length. The latter feature simplifies control and programming. In any event, the six bit byte does not represent a good choice because its length in bits is not a power of two. Apparently it would require more circuitry to handle a sequence of byte sizes; such as 1, 4, 6 than it would 1, 4, 8. A possibility which might be considered, however, is to permit some treatment of a two-bit byte in order to facilitate handling of six-bit characters from other systems (such as 705). There need not be the ability to treat as a whole a six-bit byte. There need be only the limited power to convert six-bit bytes to four or eight by programming successive applications of two-bit and four-bit operations. This point of view may ease the requirements assumed in the previous section with regard to the Exchange. The hard decision is how to handle decimal quantities in a system which permits alphabetic information. Excluding the six-bit byte, the remaining possibilities are:

- 1) four-bit bytes
- 2) eight-bit bytes
- 3) four or eight-bit bytes

The four-bit representation alone is the most economical of space, but leads to difficulties when fields are mixed containing both numbers and characters. Since it is not always possible to anticipate the location within field of each type of information it is not feasible to represent numbers by four bits and letters by eight-bits within the same area of information.

The use of eight bits for decimal numbers clears up the mixed field problem, but wastes memory space and memory accesses. It is probably indefensible from the economy standpoint alone.

What remains, and looks most desirable, is to regard decimal numbers as four bits long when they are in use arithmetically, and as eight-bit bytes when they form part of a mixed field. Also, for certain input-output uses the conversion from four to eight will be required. This will be discussed further in a subsequent section on input-output.

4. ECONOMY OF STORAGE

Economy of storage in representing characters is as important in auxiliary storage (tape, disc, etc.) as in memory. We may therefore say, first, that economy of storage is a system consideration and, second, that there is no particular virtue from the standpoint of space in adopting one code in memory and another in auxiliary storage.

It appears that in commercial applications there exists a preponderance of purely numeric information over alphanumeric. According to a recent survey by Mr. W. Heising, for example, the ratio is approximately 4 to 1. Thus, we may order the various combinations of fixed byte sizes on the basis of the minimum number of bits necessary to represent A9999:

<u>Byte Sizes</u>	<u>No. of Bits</u>
4 for num. 6 for alpha	22
4 for num. 8 for alpha	24
6 for both	30
8 for both	40

This table clearly indicates the superiority of having 2 byte sizes rather than one from the storage viewpoint. Further, the 4 and 6 bit combination is only slightly superior to the 4 and 8 bit combination.

The more byte sizes a machine possessed the more cumbersome a process it becomes to print out a region of memory, either by a general or by a special print program. There are many ways in which changes in byte can be handled; all the methods considered, however, require some additional memory space for information defining boundaries of strings of bytes of uniform size. It is true that memory maps can be kept in auxiliary storage until required and that maps may be abbreviated by taking advantage of recurring patterns where various byte sizes are intermixed in a regular manner. We may also observe that the printing problem is quite cumbersome anyway due to the existence of different formats for instructions, index words, logical and fixed point binary data, floating point data, etc. Even so, with every new byte size the cumbersomeness is increased, and so is the demand on memory space for boundary information.

In our judgment this demand on space is more than offset by the economy gained for each character by using a different byte size for pure numeric from that used for alphanumeric.

If any byte size is chosen which is not a divisor of the word size (64 bits) then some memory space will very likely be wasted. For example four bits are left over when ten-six bit bytes are packed into a sixty-four bit word.

The amount of wastage reduces the space advantage enjoyed by the four and six bit combinations over the four and eight to the point where other considerations must determine our preference for one or the other.

To summarize our position with respect to economy of storage, we maintain that a decided advantage is to be gained from using two byte sizes (in addition to straight binary) one for numeric, the other for alphanumeric, but there is little to choose between the four and six bit combination on the one hand and the four and eight on the other.

5. SEQUENCING OF ALPHABETIC AND NUMERIC INFORMATION

The problem of choosing suitable byte sizes was considered in relation to the following requirements:

- 1) The code adopted for alphanumeric data must comply with present standards of ordering based on the type 089 collator. Starting at the low order end the sequence should be as follows:

Blank
Special characters (as on Type 407)
Alphabet A to Z
Digits 0 to 9
Sentinels (special characters which can be used to make boundaries between different types or areas of information)

The bytes representing each character must be capable of being ordered as binary numbers in the sequence stated above.

- 2) Each alphanumeric character must be uniquely represented by its corresponding alphanumeric byte; more specifically the code should be devoid of intra code "case" or "shift" characters* which change the meaning of subsequent bytes but not the byte size. A necessary result of this requirement is that the alphanumeric byte must be at least six bits in length. The six and eight bit byte sizes conform to these requirements.

The above requirements leave a considerable amount of freedom in defining a structure for an eight-bit code. It is felt that the following proposals concerning the structure of an eight-bit code represent desirable objectives:

- 1) The four-bit purely numeric representation of any digit should be obtainable from its eight-bit alphanumeric representation merely by dropping the four high order bits.
- 2) Twenty seven consecutive (or uniformly spaced binary configurations should be set aside for the alphabet, the extra one over the usual twenty six being assigned to represent the 0-1 punching combination in a card column which falls between R and S in alphabetic sequence. (In this way compatibility with the 705 will be simplified).

* Intra-code case or shift characters alter the meaning of subsequent bytes while staying within the code. Intercode case or shift characters indicate a complete change of code (possibly also by byte size).

- 3) Binary configurations should be reserved at the top end of the code (and, if possible, at the bottom end too) for sentinels. Such characters are of considerable value in automatic and some non-automatic programming.

6. RECOGNITION OF BYTE SIZE

If the proposed set of byte sizes (one, four, eight) be adopted, it follows that there will be instances in system operations when it will be necessary to distinguish among them in order that the information they represent be treated properly. For example, shall two words be added as binary numbers or as coded decimal numbers? Shall a stream of bits reaching an output device be interpreted four at a time to indicate digits, or eight at a time for letters and other characters? If the memory is being "printed out", how shall the contents of memory be reproduced so as to be meaningful to the programmer trying to debug a program?

There are two ways in which to treat this problem. One is to assume that the data (instructions, tables, records, etc.) contains no useful clues to its identity and that its proper utilization depends strictly on prior knowledge of the programmer about his material. That is, he instructs the adder to perform a decimal addition because he knows that data is decimal. He selects the printer and instructs it to operate in a specific byte size mode, to accept either four bits per type wheel or eight bits per type wheel. Such control information is not part of the data stream, but rather inherent in the instructions controlling the output device. To operate in this manner would require that the printer operate under computer control and not as part of an off-line tape to printer operation.

Another approach to the problem is to assume that the data is self identifying. That is, at the required interval the data tells the system what byte size it represents. This identification may be by the byte, by the field, by the word, or by the area. Each of these possibilities has its applications:

By the byte: for streams of alphabetic and numerical characters going to a punch or a printer

By the field: for streams of data entering the arithmetic unit for comparison

By the word: for numbers going to the adder

By the area: for the printing out of memory

Unfortunately, no single one of them is sufficient for all uses, and in fact, where some information in binary is likely to be part of the data stream there exists the possibility of false indications whenever the implicit indications of byte size are not in predictable locations.

On the other hand, there are instances in which implicit indicative information may be necessary unless other measures are taken. For example, in the case of streams of characters going to the printer, there will have to be some sort of indication if there is to be a mixture of byte sizes and no interruption of the stream for control purposes. A possible solution is to employ "shift codes", provided that binary information is prohibited.

We believe that, in general, implicit indications (self-identifying devices) provide a poor solution to the problem of size of byte recognition. Whenever the program has control, it seems best to have the operation define the byte length. In going to external units, it seems best to present a uniform size to the unit. Thus, where there may be alphabetic characters in the stream, all bytes should be eight bits long. For memory print out, some sort of memory map will have to be provided separately from the data--or else let the memory be printed with each line repeated three times: once in binary (octal preferably), once in decimal digital, and once in translated eight-bit form. Some major decisions remain to be made in this area.

7. INPUT-OUTPUT EXCHANGE

There is no internal byte size which will satisfy concisely all external equipment. Further, in many cases a non trivial translation may also be involved (adjoining and dropping zeros are considered trivial translations).

It looks at present, as though the on-line card machines (reader, punch and printer) will be operated by means of row by row card images supplied to or from the memory, translation of these images being carried out internally. Preliminary analysis of this translation problem indicates that it is as easily handled with four bit and eight bit bytes as with six bit bytes. However, the problem is still under study. Regardless of whatever aids are built into the computer or card machines to simplify translation, it is highly desirable to retain the availability of card images with all the flexibility they imply.

If an eight bit byte is adopted internally as well as in the Exchange it becomes extremely desirably to adopt eight bit magnetic tape units as the commercial Harvard standard rather than six bit units. (NOTE: in each case we are speaking of information bits and excluding consideration of checking bits for the time being). The system must still be able to communicate with six-bit magnetic tape units in order that it be tape-compatible with the 704, 705 and off-line peripheral tape equipment.

Some of the implications of communicating with eight-bit and six-bit tapes are revealed in the diagram on page seven.

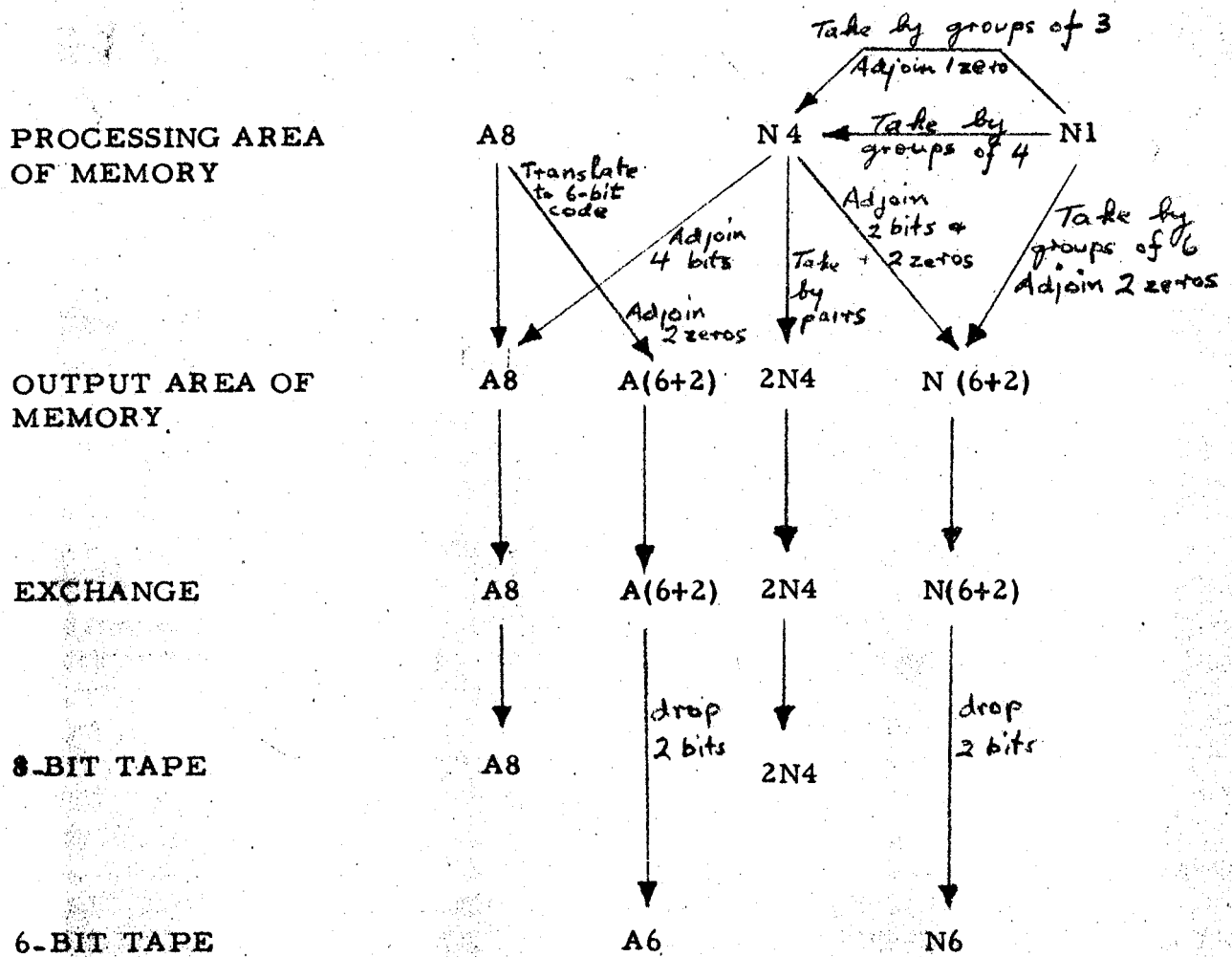
The symbols used may be understood from the following examples:

2N4 denotes two numeric bytes each four bits long; A8 denotes one alphanumeric byte eight bits long; A(6+2) denotes one alphanumeric byte eight bits long, but expressed in a six-bit code with two dispensable bits adjoined.

It is felt that if for any reason eight-bit magnetic tapes were not made available the desirability of using an eight-bit byte internally would be seriously reduced.

The low speed RAMAC, if it is to be connected to the Exchange at all, should be set up for operation with eight-bit bytes (where once again, we remind the reader

Fig 1: MANIPULATION OF 1, 4, AND 8 BIT BYTES FOR RECORDING ON 6 AND 8 BIT MAGNETIC TAPES



The process indicated above is reversible providing the programmer knows in advance the coding of the input from magnetic tapes. This is a reasonable assumption to make.

that the redundancy bit has not been counted: hence, it and a space bit, if necessary, would imply a total of ten-bits per character). The reason for desiring eight-bit operation of the low speed RAMAC is identical with that for magnetic tapes; namely to obtain concise representation in auxiliary storage of all 256 possibilities which an internal eight-bit byte implies.

8. WORD, BYTE AND BIT ADDRESSING

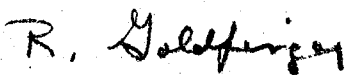
It is assumed that we wish to address each bit in memory and that the addressing scheme is itself binary. It seems desirable that bit addressing and word addressing be so related that a simple count of one transforms the address of the last bit of word N into the address of the first bit of word $(N+1)$. This is possible only if the word length is of the form 2^p where p is an integer. This condition is, of course, satisfied in the machine under consideration ($p=6$).

Similar remarks apply to byte addressing. Thus, byte sizes should also be of the form 2^p (where p is an integer). The four and eight-bit-bytes satisfy this condition; the six-bit byte does not, and this is considered a drawback of this particular size.

9. CONCLUSION

1. The four and eight-bit byte sizes represent the best compromise for the Commercial Harvest machine between economy of storage and ample capacity of representation of extended character codes.
2. Some translation and identification problems are going to be encountered in having memory communicate with external units or other machines. However, it is felt that each such difficulty is capable of solution in a reasonable way without the need for exceptional recognition or translation devices. However, the use of the three byte sizes (1, 4, 8) will necessitate a greater reliance on programmed editing.


E. F. Codd


R. Goldfinger

EFC:RG:jv

cc:Mr. S. W. Dunwell
Mr. B. L. Sarahan
Mr. J. C. Gibson
Mr. W. Buchholz