BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W. WASHINGTON, D. C. 20024

SUBJECT: The Bellcomm FORTRAN I/O Package
Case 803

DATE: February 13, 1969

FROM: W. M. Keese

TM-69-1032-1

TECHNICAL MEMORANDUM

Introduction

On a computer as fast as the UNIVAC 1108, a large proportion of all programs are I/O bound. That is, the computations performed on data frequently can be done in less time than that required for reading in the data and writing out the results, so the computation process must wait upon the Input/Output process. Thus, the time and expense of running a program is frequently determined more by the efficiency of the I/O process than that of the program.

Computer programs written in the FORTRAN language (as most of Bellcomm's are) do their input and output of data through the use of a set of library programs, collectively called the FORTRAN I/O package. This package for the 1108 EXEC II has been rewritten at Bellcomm in order to obtain higher efficiency. It obtains this efficiency in two main ways: buffering and item packing (of which the latter is the most important in Bellcomm usage). This package effects I/O only for mass storage devices (tapes, fast drum, FASTRAND), not for unit record devices (printers, card equipment).

Buffering

The I/O (Input/Output) process is said to be buffered if it occurs concurrently with computation so that the latter process need not halt for the I/O operation.

For input, this means that the input of data must be initiated into a buffer some time before the data is to be wanted. Then, when it is wanted, the buffer can be copied into the data area with no delay for the relatively lengthy I/O process. This form of initiating input ahead of time is sometimes called buffering ahead. The Bellcomm package contains provision for buffering ahead in a physical sense, but this effect is currently inhibited.* It does, due to item packing, buffer ahead in a logical sense.

For buffered output, the data is copied into one or more output buffers and the computational process is allowed to continue simultaneously with the output of these buffers to the external device. The Bellcomm package always utilizes buffered output.

Item Packing

The FORTRAN language provides facilities for writing and reading logical entities called records. A logical record is the indivisible unit in which the programmer can do I/O. Although he can specify the pieces of data which constitute such a record, he must read it or write it all at once; it can not be read or written piecemeal.

Similarly, many I/O devices (such as magnetic tapes) are written in physical records, this being their indivisible unit of communication. Physical records are frequently called blocks.

The most simple form of I/O package allows a direct correspondence between these two forms of records. However, this can be undesirable in a number of ways. If records are very short then: 1) they can be mistaken for tape noise; 2) effective tape utilization can be very low (e.g., 1-2%); 3) speed is greatly reduced due to excessive expenditure on device access times.

Any buffering scheme implies that there must be some maximum length of a physical record (the buffer size). Thus, long logical records must be partitioned into shorter items which are issued as several blocks (physical records). UNIVAC refers to this as blocking, although this word is commonly reserved for the more complete process which we will call item packing. This limited form of blocking imposes a slight speed penalty on the I/O of long logical records, but this is minimal if the block length exceeds approximately 200 words, and the reliability of the process is enhanced. Further, the speed advantage of the buffering usually overwhelms the blocking loss.

* Due to FASTRAND handler deficiencies.

The aspect of blocking just described limits the length of a physical record. A far more important aspect, herein called item packing, assures a minimum physical record size. In this scheme, short logical records are packed together into longer physical records.

As done at Bellcomm, the minimum block size is equal to the maximum block size, 256 words. This is called a fixed block size algorithm. In such, it is occasionally necessary to waste a little space in order to fill out a block, but considerable programming complexity is avoided.

Definition of an Item

In the Bellcomm package, a tape may be viewed in two different ways: firstly, it is a set of data words partitioned into a sequence (ordered set) of physical records; secondly, it is a set of data words partitioned into an ordered set of logical records. Given any two partitions of a set, there exists a unique partition called the crudest common refinement of these partitions. The items are merely the elements of this crudest common partition (with some control words). They are the longest sequence of data words such that; 1) the tape is a sequence of items; 2) each physical record is a sequence of items; and 3) each logical record is a sequence of items.

This is not the only possible definition of an item, but it is mathematically the most simple one, and it has the implication that items do not span physical blocks (although logical records do). This implication allows more data recovery from tapes which develop bad spots.

Generation of Items

When the programmer commands the writing of a logical record, buffer space is saved for a control word and his data words are inserted in the buffer until either there are no more data words or only 2 words remain in the buffer. A checksum is put immediately after his data. An appropriate control word is generated and placed immediately in front of his data and immediately after the checksum. His data, together with the three added words, forms an item. If the buffer is full, its output is begun. If there are more data words to write, another buffer is obtained, and the process continues (saving space for a control word, etc.). Computation continues while any full buffers are written out.

Compatibility

Tapes written by the Bellcomm I/C package are different from those written by the UNIVAC package. Further, the UNIVAC package allows for tapes written in two different formats, called binary and formatted. (Most tapes are of the binary type). The Bellcomm package uses the format described above for both kinds of output.

The Bellcomm package can read the old style binary tapes with no special provision. It can write an old style tape or read an old style formatted one only if the package is informed that the tape(s) in question are old style ones. This can be done by "CALL X2TAPE (N1, ...,Nn)" where the "Ni" are the tape numbers of the old style tapes.

Relative Efficiency

There are two measures of efficiency for an I/O package: packing fraction and speed. These must be measured differently for tape and FASTRAND. In either case, there is no simple formula for speed. The average speed of an item packing package is a function of record lengths and their distribution in time. If, for instance, I/O is produced in sufficiently short and sufficiently separated bursts, then there can be a complete overlap of I/O and computation, so that I/O cost approaches 0. Conversely, if a run is completely I/O bound and deals with very large records, then there is no advantage whatsoever to the new I/O package. The new I/O package always shows its worst comparison in the case of the completely I/O bound run.

For magnetic tape (I/O bound case), speed varies quite directly with the packing fraction (expressed, say, as number of data words/inch). The packing fraction of the Bellcomm package varies from 19 times as good as that of the UNIVAC package (for 0 length records) to the same. It is always at least as good. Roughly, it is 16 times as good for 1 word records, twice as good for 21 word records, and essentially identical beyond 250 word records. Inverse speeds are shown in Figure 1. For the non I/O bound case, speed improvement becomes much better than packing fraction improvement.

For FASTRAND (I/O bound case) this correlation does not hold. Again, the Bellcomm packing fraction is always at least as good as the UNIVAC, but less so. On this device, speed varies inversely as the number of device accesses necessary, so the speed improvement for the Bellcomm package is much more notable. Times to write 1,000 records are plotted as a function of record length in Figure 2. The dual step nature of the UNIVAC curve (due to lack of buffering and periodic accessibility) precludes any simple formula for comparison. In the significant cases noted below, however, the speed improvement on FASTRAND is over twice the improvement on tape.

Although relative efficiency is a function of record length and I/O distribution, two significant cases can be singled out:

It appears that appreciably all work with formatted files at Bellcomm deals with card images (14 words long). For these, on FASTRAND the Bellcomm package shows 1/2 again the packing fraction and a speed improvement of 8 to 1. On tape, the packing fraction and speed both improve by 4 to 1.

An (unpublished) survey of several thousand user tapes at Sandia revealed the average binary record length to be 31 words. For such records, the Bellcomm package shows a packing fraction improvement of 1/3 and a speed gain of 7 1/2 to 1 on FASTRAND. On tape, the packing fraction and speed both improve by a factor of 2.

Actual Running Times

Individual programs can be demonstrated which give the Bellcomm package a relative speed advantage ranging from 0 to infinity*, but such selected figures are meaningless. The only statistically meaningful figure would come from capturing, say, a week's runs and running them with both packages. Unfortunately, this would be prohibitively expensive, and the progressive nature of many runs (changing of data and/or programs) makes unauthorized rerunning impossible anyway. Further, associated changes to utility routines which were put in with the Bellcomm I/O package (in both the library and user's program files) make duplicate runs impossible except in selected cases.

* One phase of a production program went from a measured 18 seconds to a measured 0.
This kind of effect can occur because the Bellcomm package is able to do some redundant I/O operations entirely in the buffers (e.g. WRITE-BACKSPACE-WRITE).

Individual figures do exist on our two most used production programs, however.

BCMASP is reported to run in about 60% of the time that it did, an effective speed increase of 66%.

BCMRET is reported to be running at least 4 times as fast as it did under the old package.

The author feels that the average program execution at Bellcomm is approximately twice as rapid since the introduction of the new package. Assuming that between 30 to 60% of our computer time is spent in execution, this saves 15 - 30% of our running time resulting in a throughput increase of 20 to 40%. In a normal, production-oriented FORTRAN computer operation, the throughput increase should run 70 to 90%.

[illegible]

1032-WMK-dmu W. M. Keese

Attachments
Figures 1 & 2

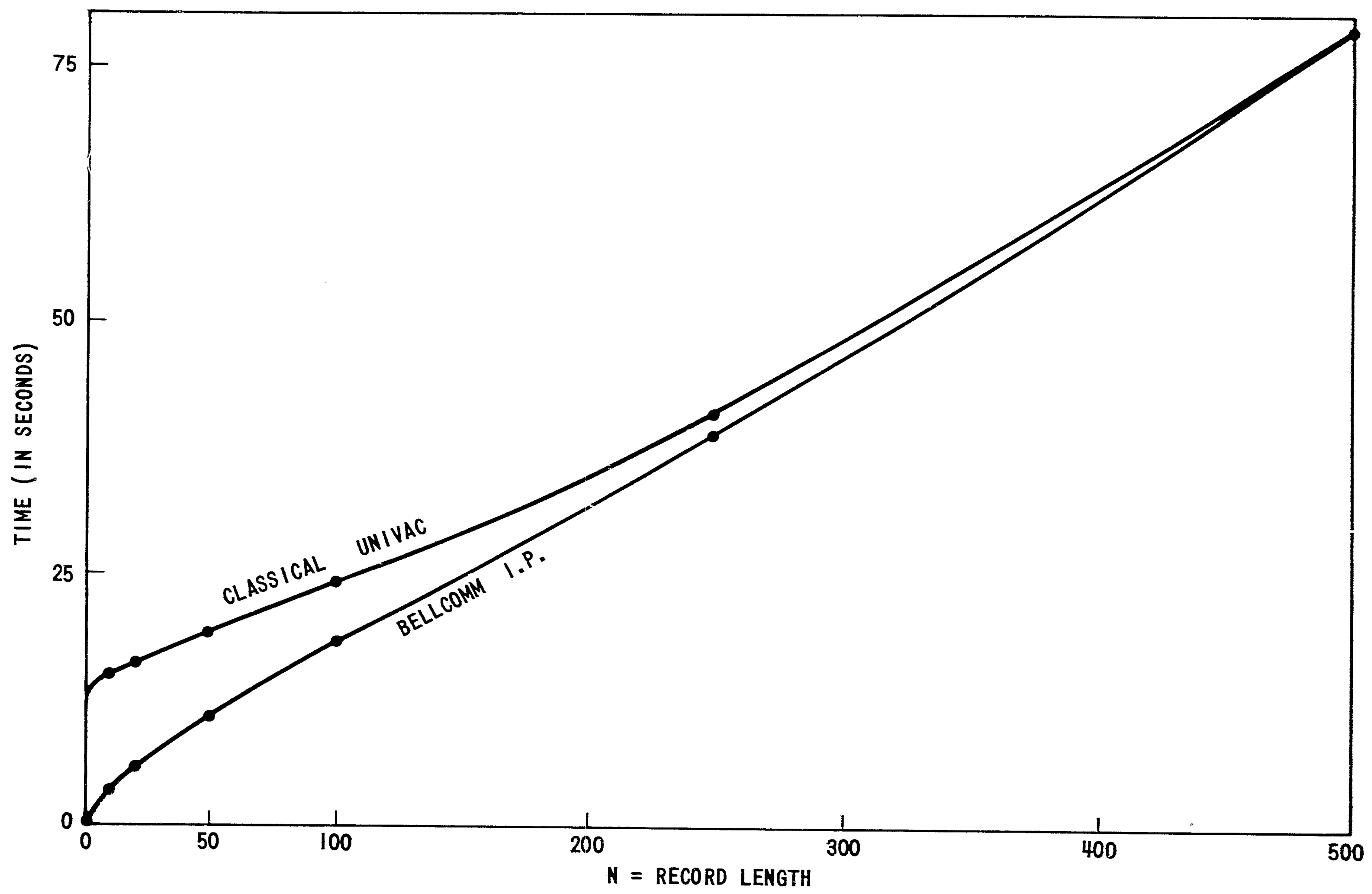

FIGURE I - FORTRAN I/O SPEED TESTS ON TAPE (556 BPI) TO WRITE 1,000 RECORDS OF N WORDS EACH (NO COMPUTATION, WRITE TIME ONLY)

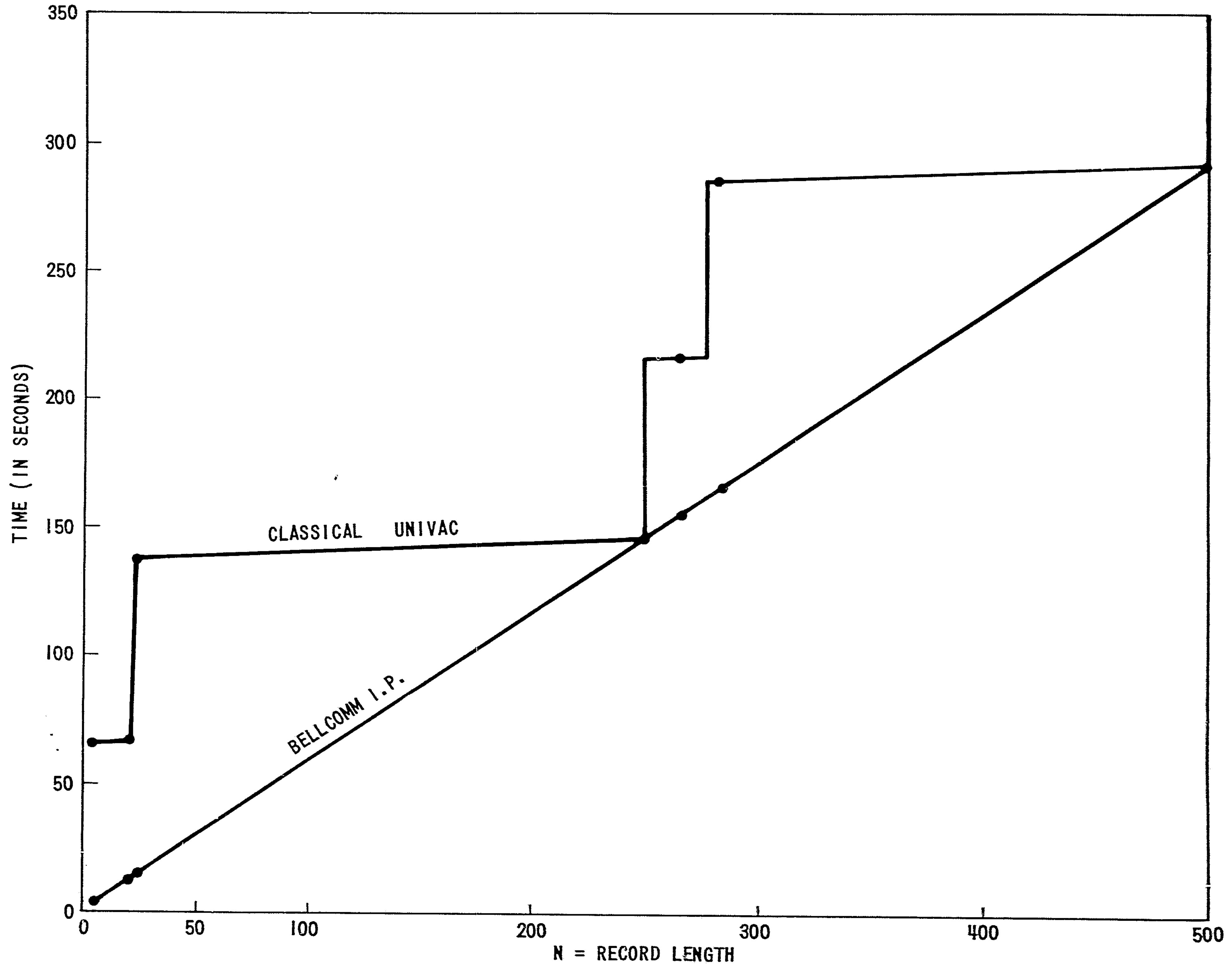

FIGURE 2 - FORTRAN I/O SPEED TESTS ON FASTRAND TO WRITE 1,000 RECORDS OF N WORDS EACH (NO COMPUTATION - WRITE TIME ONLY)