Observations on Microprocessors and Computing Efficiency

C. C. Klimasauskas and J. W. Layland
Communications Systems Research Section

This article presents results of a continuation of our previous study of computing efficiency in DSN-related tasks. The two task models considered here are manipulation of man-readable character-string data, and a very simple process variable monitoring operation. Several currently available minicomputers and microprocessors are compared. The principal result is that the microprocessors appear considerably more cost-effective than the more powerful machines for the simple repetitive tasks, but that the opposite is true for more complex operations.

I. An Introduction to the Problem

In the last year, a number of large-scale integrated-circuit (LSI) microprocessors have become available with the promise of more to follow in the near future. At present there is little understanding, or experimental data, to determine which tasks they are best able to perform. This article considers some of the complexities of the general problem of selecting a particular processor/implementation, specifically in the area of processor trade-offs for different tasks. It is a continuation of an earlier study done comparing minicomputer implementations for performing DSN-related computer activities (Ref. 1).

II. Time–Storage Costs

This section considers the cost effectiveness of the implementation of a macroprocessor called STAGE-2 (Ref. 2). STAGE-2 manipulates character-string data supplied from one input/output (I/O) device and emits character-string data to another device. Its requirements in terms of machine capabilities should be similar to the text manipulation performed by a DSN subsystem controller in conversing with its operator.

STAGE-2 is a language-independent macroprocessor written in a machine-independent language called FLUB. This machine-independence makes it relatively easy to develop alternative implementations on one computer, or many computers, and compare their performance directly. The operation statistics were gathered through a special instrumented version of STAGE-2. Three alternative implementations were studied for the Sigma 5: The first was a straightforward implementation, packing all the information contained in a FLUB word into a single 32-bit Sigma word. The second was an implementation which minimized execution time. In this implementation, FLUB data were stored in 64-bit double words. The third was an interpretive implementation to minimize storage. Our motivation for this implementation comparison may be

found in the work of Savage (Ref. 3), who showed that a bound exists to the minimum time–storage product for any well-defined process. Storage in all implementations included the STAGE-2 instructions and 4000 FLUB memory words. These implementations were compared in two ways: One comparison ignored processor cost, and considered only the resources of storage and execution time. The other included the cost of the processor in the storage requirement. Processor storage equivalent was computed from the following:

$$\text{Processor storage equivalent kbytes} = \frac{\text{Processor cost at initial marketing (\$)}}{\text{\$ for 1 kbyte of processor storage at initial marketing}}$$

The storage–time diagram for the implementations is shown in Fig. 1. The dashed lines represent a constant storage–time product. The single-word implementation appears cheaper than the double-word implementation when the processor cost is ignored. When the processor cost is included, the opposite is true. In both cases the interpretive implementation is always costlier. This is due to the high time overhead in decoding the FLUB instructions as compared to the savings in storage. However, programs such as high-level language translators may benefit from interpretive implementation, being tasks in which the functions performed are complex, storage large, and the instruction-decoding overhead small. Processor cost is clearly significant here, and is included in all following material.

The two faster implementations fall roughly upon a constant time–storage product line. The interpretive implementation would clearly be a mistake to use. Physical constraints of the task have presented another bound to implementation alternative. We observe that other constraints, such as a requirement to force-fit a task onto predetermined hardware can result in extremely inefficient use of the computing resources.

III. STAGE-2 Processor Comparisons

STAGE-2 was implemented in fact, or hypothetically, on a variety of minicomputers and microprocessors. Processor selection was based on information availability. From manufacturer's published data, instruction counts from the instrumented version of STAGE-2, and knowledge of implementations, the execution time and storage requirements to perform a specific macroprocessing task were computed for each processor. These data are plotted in Fig. 2. For each processor, the processor storage equivalent was considered part of the storage requirement.

Of historical note, the data on the 8080 were not available when Fig. 2 was first plotted. Without the 8080, it appeared that STAGE-2 was just too complex a job to be implemented on a microprocessor. However, the 8080 is approximately on the same storage–time line as both the PDP-11/20 and Sigma 5. The 8080 implementation is well over 10 times faster than the 8008 (though about 10 times faster in raw processor speed) and requires considerably less program storage. This results from the availability of 16-bit data manipulating instructions on the 8080. These more powerful data manipulating instructions, a faster processor, and more powerful addressing modes compound storage–time savings.

It is worth noting that each of these processors lies within the storage–time products of 1200 kbyte-min and 2400 kbyte-min. This implies that as resources for performing this task, they are essentially equivalent in cost. This may be a phenomenon of implementing a machine-independent task equally poorly on a variety of machines. This warrants future consideration since the trend in computing is away from machine-dependent features and toward, as much as possible, machine-independent algorithms and implementations. For example, Professor Per Brinch Hansen at the California Institute of Technology is working on a transportable multiprogramming operating system written in concurrent PASCAL, soon to be completed on the PDP-11 (Ref. 4).

In these comparisons, it is essential to realize that a salient feature of STAGE-2 is that it does not use indexing nor indirect addressing as such. A task which extensively uses indexing would increase the 8080's and 8008's storage–time product over its competitors, possibly eliminating them from consideration. Also, any task which required 32-bit arithmetic operations would move the XDS 930, PDP-11's, and 8080 out of the competitive region. The same is true of floating point tasks.

This exercise indicates that microprocessors may be competitive with minicomputers for performing simple tasks which utilize direct addressing and some 16-bit data manipulation (for example, emulation of the FLUB machine). At the same time, it is clear that we can find some task which is complex enough to make any specific microprocessor (or minicomputer) uneconomical with respect to a more complex processor.

IV. System Variable Monitoring, Processor Comparison

Within a DSN tracking station, computers are used to monitor an ever increasing number of variables. This enables operators and engineers to identify real and potential failures immediately, to isolate them, and to identify potential bottlenecks in performance. As tracking and flight control complexities increase, variable monitoring is expected to increase in sophistication and scope to meet the new demands. One approach to this problem is to employ a station "master" monitoring computer (MMC) which interfaces to a number of variable monitoring systems (VMSs). Each VMS would be responsible for reporting to the MMC those variables which exceed predetermined limits. This might be done by treating each VMS as a peripheral device of the MMC connected to a channel multiplexer. Through the channel multiplexer, the VMS would periodically test each variable. Figure 3 illustrates the task graphically.

The VMS was hypothetically implemented on a variety of microprocessors and minicomputers. Though better implementations may exist for each processor, an attempt was made for each implementation to maximally utilize speed and processing power. Execution time and storage requirements were computed from manufacturer's specifications. Storage for each processor implementation includes the processor equivalent storage. For purposes of comparison and from practical considerations, a sample interval of 20 ms was assumed. Figure 4 illustrates cost as a function of channel capacity. Whenever the sample interval could not be achieved with a single processor, multiple-processors were employed.

In this job, the 8080 microprocessor and even the 8008/8008-1 are far more cost-effective than any of the minicomputers. At large channel capacities, however, the Modcomp II is competitive with the 8008. The 8080 has about 10 times the capacity of the 8008, because it is a faster processor with more powerful addressing modes and data manipulating instructions. In this task, the high initial cost of the minicomputers is not offset by their faster processors and more powerful instruction repertoires. This small, repetitive task with a data width of 16 bits seems to be an ideal job for microprocessor implementations.

Of possible technological interest is that in the asymptotic limit system cost appears to be a function of date of introduction. Notice that the Sigma 5 and PDP-11/20, the Modcomp II and 8008, the 8008-1, and the 8080 are from increasingly more recent technological eras, and are correspondingly less expensive. This is emphasized by the task which is minimal, fits each processor comfortably, and is tailored to maximally utilize processor power.

The VMS is a task which well fits the capabilities of the microprocessor. The straightforward addressing required, short data word, and low initial cost make it much more attractive than its faster more sophisticated predecessors, the minicomputers.

V. Remarks on Storage Equivalent

Throughout this article, storage has been used as a unit of processor cost. Principally we hoped to minimize the effect of manufacturing processes and technology by assuming that both storage cost and processor cost were directly proportional to technological and manufacturing developments. If this, in fact, were true, the resulting measure of processor storage equivalent should make the comparisons significantly less dependent on the state-of-the-art and more dependent on processor architecture and power. Also, prices for processors and storage are continually changing, and vary enormously between sources. As such, storage equivalent provides a uniform well-defined standard on which to base processor cost.

The principles used in this study can easily be adapted to a direct cost comparison, in which both storage and processor costs are expressed in dollars.

VI. Summary

Though the results of this study are limited in scope and not firmly conclusive, we would like to make three observations. First, processor cost must be included in evaluating processor-implementation performance. Secondly, microprocessors appear competitive in jobs with short data items and simple addressing requirements. Finally, microprocessors appear best-suited for simple repetitive tasks, such as DSN variable monitoring.

References

1. Layland, J. W., and Klimasauskas, C. C., "A Myopic View of Computer-Based System Design," in *The Deep Space Network Progress Report*, Technical Report 32-1526, Vol. XIII, pp. 154–167, Jet Propulsion Laboratory, Pasadena, Calif., Feb. 15, 1973.

2. Waite, W. M., *Implementing Software for Non-Numeric Applications*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1973.

3. Savage, J. E., "Computational Work and Time on Finite Machines," *J. ACM*, Vol. 19, No. 4, pp. 660–674, Oct. 1972.

4. Hansen, P. B., *DEAMY, A Structured Operating System*, CIT-IS Technical Report 11, California Institute of Technology, Pasadena, Calif., Mar. 1974.

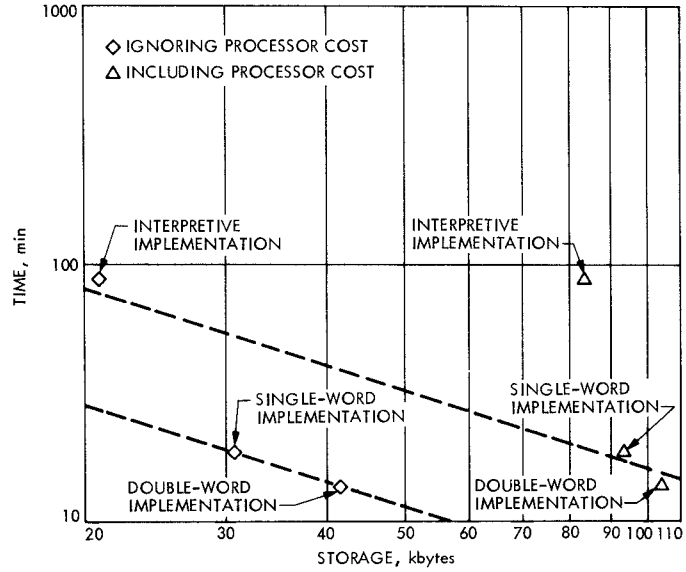


Fig. 1. Storage-time diagram for STAGE-2 implementation on Sigma 5

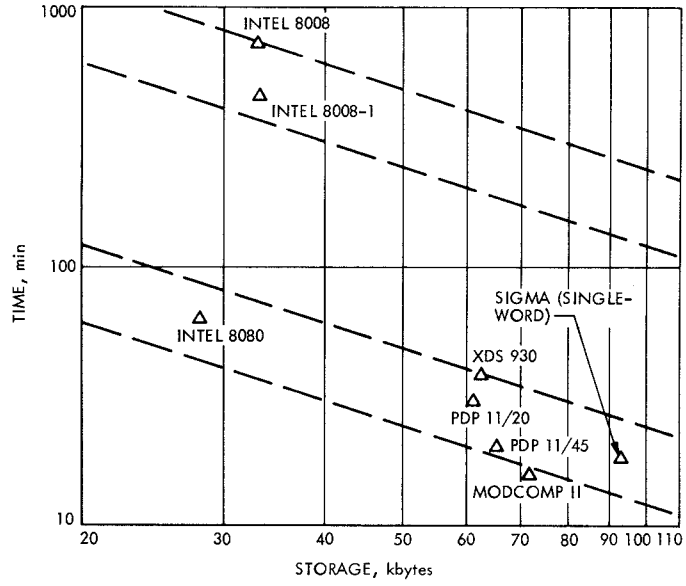
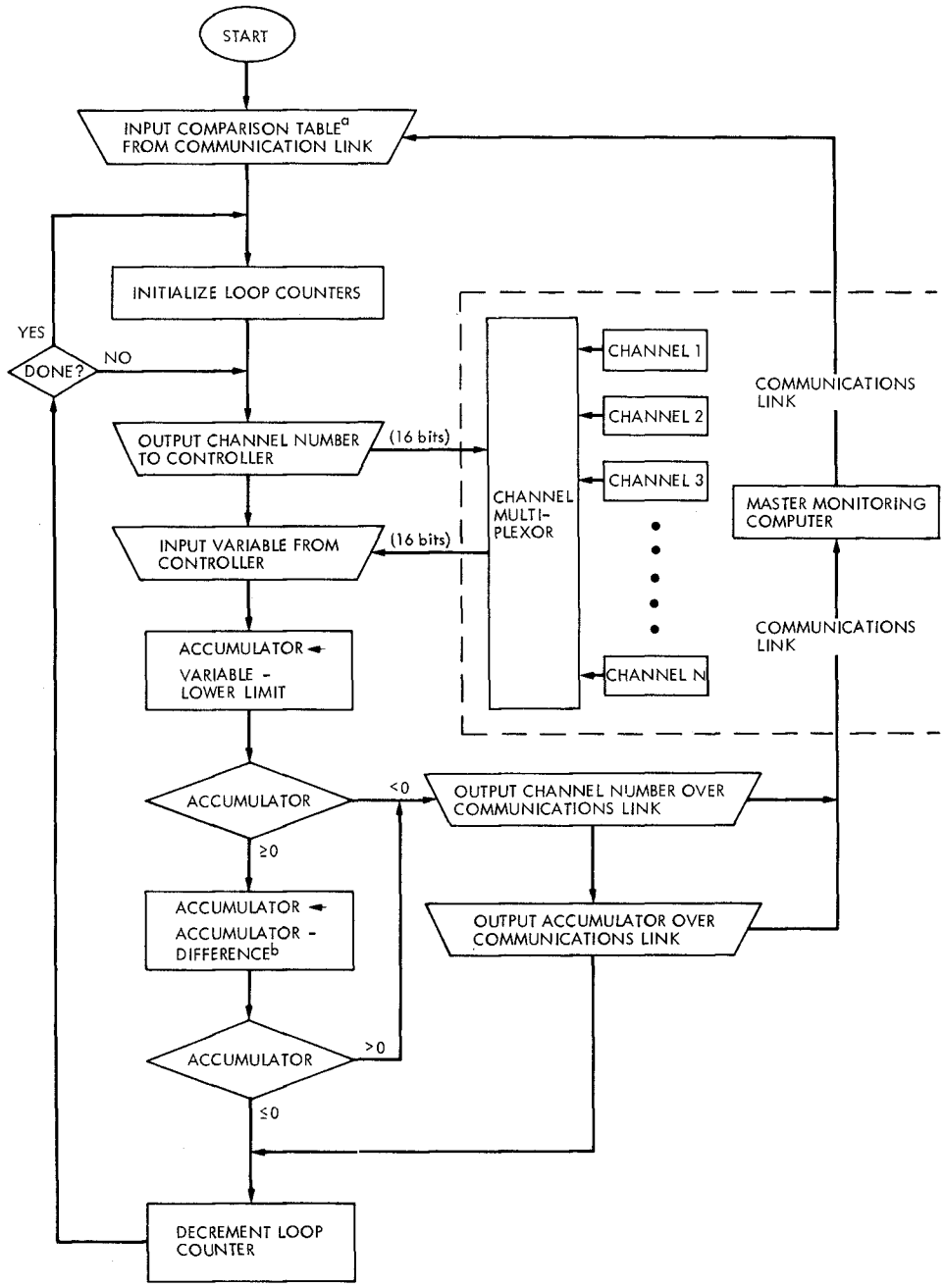


Fig. 2. Storage-time diagram for STAGE-2 on several different processors



[a]TABLE IS IN STORAGE-IMAGE FORMAT SPECIFIC TO TARGET PROCESSOR

[b]DIFFERENCE IS THE QUANTITY (UPPER LIMIT - LOWER LIMIT), WHERE UPPER AND LOWER LIMIT ARE, RESPECTIVELY, THE MAXIMUM AND MINIMUM ACCEPTABLE VALUES FOR VARIABLE

Fig. 3. Flowchart of the variable monitoring system

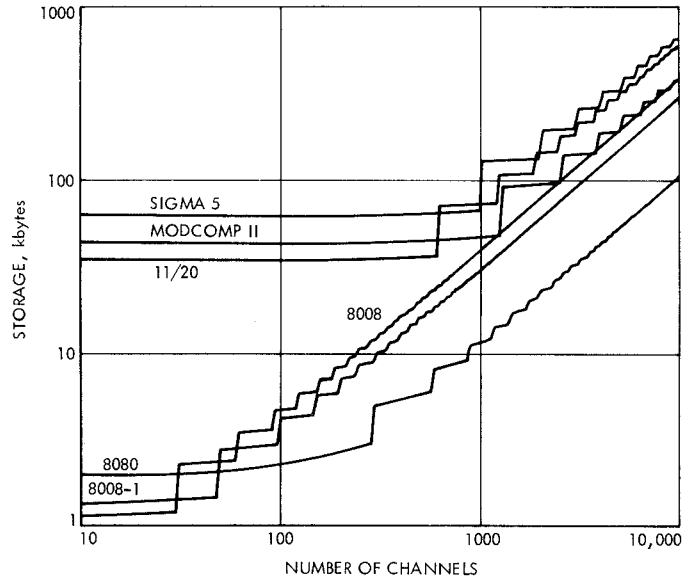


Fig. 4. Variable monitoring system cost as a function of channel capacity and processor