GENERAL ELECTRIC
COMPANY

13430 NORTH BLACK CANYON HIGHWAY, PHOENIX, ARIZONA 85023 . . . TELEPHONE 941-2900

COMPUTER
DEPARTMENT

DEER VALLEY PARK PLANT

May 12, 1966

Mr. Jerry Saltzer
Project MAC
545 Technology Square
Cambridge, Massachusetts

Dear Jerry:

Enclosed is a copy of our proposal for on-line performance monitoring in the 645. As it stands, this report is a preliminary attempt to establish both hardware and software measures. The approaches outlined therein require alterations in both the hardware and software.

We would appreciate your comments and suggestions.

Sincerely yours,

[illegible]

J. Shemer
Sr. Systems Engineer
Modeling and Analysis Subsection
Systems and Processors Operation

JS:cm
enc:

SYSTEMS DESIGN AND ANALYSIS MEMORANDUM

CATEGORY: 1- 645 ANALYSIS
c) SYSTEM STUDIES

REPORT NAME: " PROPOSAL FOR ON-LINE PERFORMANCE MONITORING OF G.E. 645"

DATE: APRIL 20, 1966

SERIAL NUMBER 29

AUTHORS [illegible] Apr 20, 1966
I. EPSTEIN

J.E. Shemer April 20, 1966
J.E. SHEMER

G.A. Shippey by J.E.S. April 20, 1966
G. A. SHIPPEY

ORIGINATED BY: MODELING AND ANALYSIS SUBSECTION
SYSTEMS AND PROCESSORS OPERATION

APPROVALS [illegible] April 25th 1966
J. DOBBIE

INTRODUCTION

There is wide agreement that on-line performance measurement and program tracing will be an essential activity on the MAC-BELL 645 Systems as soon as they become operational. In Reference 1, J. F. Gimpel of Bell makes the following comments:

> " Performance monitoring activities in Multics will probably serve at least the following purposes
>
> (1) System debugging
>
> (2) Compiling statistics to improve software strategies (e.g., scheduling algorithm, paging algorithm).
>
> (3) Real-time statistics gathering to affect parameterized algorithms.
>
> (4) On-line data gathering for users (provided safeguards exist).
>
> (5) Off-line and on-line data gathering to affect operations (i.e., computer center) policy.
>
> (6) Long range statistics gathering to aid in the design of new machines and new software.
>
> (7) Data gathering for purposes unforseen at system design time. "

Clearly Item 6 is of vital interest to the Advance 600 Line Project Sub-section.Items 2 and 3 are vital to Modeling & Analysis Subsection, more specifically to provide driving statistics for the 645 - Multics system models.[2,3]

The purpose of this note is to supplement the purely software performance measurement proposals made in Reference 1 with additional hardware and software recommendations directed towards supplying the information in the form required by the G.E. Systems Engineering groups concerned.

It is unfortunate that hardware modifications should prove necessary so near to equipment delivery date. Three categories of monitoring facilities can be provided on computer systems, viz:

a) built in; prototypes only

b) optional extra; all machines

c) built in; all machines

In the future it is recommended that considerable thought should be given as to the correct monitoring policy to adopt during the design stage.

[illegible]

the number of segment references completed in unit time. However, this figure can also be obtained from the memory cycle rate using the AM performance index described in 2.2. The processing rate

2. SUMMARY OF STATISTICS REQUIRED

2.1 STATISTIC CATEGORIES

The most urgently required performance statistics can be grouped into three classes:

a) Associative memory performance
b) Central Processer performance versus memory interference
c) System and typical user software properties

Other classes of statistics, e.g. I/O system performance, may well turn out to be very important, but they are not discussed in this note.

2.2 ASSOCIATIVE MEMORY PERFORMANCE

The AM performance figure of greatest interest is the proportion of times that a required pointer word is not found in the AM and must be retrieved from core. This statistic is required, both as an overall average over the current program load, and also as a figure of merit for specified procedures and processes.

It would also be useful to obtain by direct measurement the switching overhead; i.e. the extra number of core memory cycles needed on average to reestablish the contents of the AM after a transfer of contents, program interrupt or fault.

2.3 CENTRAL PROCESSER STATISTICS

The processing rate in any given processer is best defined by the number of segment references completed in unit time. However, this figure can also be obtained from the memory cycle rate using the AM performance index described in 2.2. The processing rate depends essentially on the "Instruction Mix" and on core memory interference, apart from the speed of the processer itself. Hence, instruction mix and, say, I/O load on memory, should be measured together with the processing rate.

The instruction mix is valuable information in its own right for designing overlap schemes in future central processers.

The data desired from spot measurements made on processing rate should be put into a semi-empirical model (under development) of the 645 Central Processer in order to obtain the coefficients.

2.4 SOFTWARE PERFORMANCE

The software statistics needed to drive the 645 System Models[2,3] include:

a) Total number of segment references for selected procedure segments (or "subprocesses")

b) Page reference distribution function[2] for selected paging units

c) Proportion of references made by selected procedures to each of their associated paging units.

Using the processing rate conversion factor described in Section 2.3, item (a), is related to the total time spent in selected procedures or processes. This would give, for example, the percentage of computer time devoted to Core Management, a quantity directly interesting in its own right, as pointed out in Reference 1.

3. MONITORING SELECTED PROCEDURE

As noted in Section 2, it is sometimes desirable to gather statistics on the total program load on the system, and sometimes desirable to obtain statistics for selected procedures.

In order to monitor the entry to, and exit from, the selected procedure some special form of Call, Save and Return is necessary which passes control temporarily to the Monitor and Tracing (MT). However, this must be supplemented by monitoring of faults and interrupts to allow for unprogrammed transfers of control. A scheme for carrying out this task is given in Ref. 1 and would probably satisfy the modeling group's requirements.

There is a possibility, however, that statistic gathering might be desirable on a procedure which is not a segment on its own right, and is not entered via Call, Save, or Return. Provided that this procedure can be defined by a certain set of pages in a procedure segment, it might be possible to isolate the procedure for statistic gathering by tagging the corresponding PTW's in some fashion (cf. Section 6.).

4. MEASURING ASSOCIATIVE MEMORY PERFORMANCE

In Reference 4, an approach for modeling AM is outlined. This particular scheme has a number of disadvantages, namely; the input statistics required for the model necessitate a detailed statistical characterization of the software, and the time consuming Monte-Carlo simulation lacks flexibility. Moreover, the amount of detail required to obtain a statistical characterization of selected software is, in itself, a major task.

An alternate scheme is to employ two hardware counters. One counter would automatically count the total number of memory accesses, and the other would accumulate the number of this total which were pointer word accesses. Such a method affords an extremely flexible means of monitoring AM, since the contents of both counters could be stored selectively under program control during the execution of a tagged subprocess or process. Hence, the proportion of times that a required pointer word is not found in the AM can be obtained for specified procedures and also as an aggregate average for the current program load. With some additional "tracing software," the switching overhead for selected processes could be calculated through use of a timed interrupt which would periodically clear the AM to zero and return control to the interrupted procedure.

In addition, this use of counters to acculmulate memory cycles would facilitate a means for charging processes on the basis of memory cycles received and also provide statistics as to processer performance (see 5).

5. PROCESSER STATISTICS

If the two hardware counters referred to in Section 4. for accumulating the number of memory cycles are incorporated in the processer, then these counters together with a real-time clock provide the statistics necessary to calculate the processing rate. Thus, the **processor is capable of measuring its own processing rate over a wide** range of operating conditions, such as I/O load, I/O configurations, memory inter-lacing, memory priority, etc.

As for additional processer performance measures, other counters (or these same two counters) could be used to accumulate:

a) the proportion of memory cycles of a given type (RRS, CWR or RAR)

b) the number of conditional and unconditional transfers together with the number of instructions executed

c) the accumulated time spent waiting for memory response — in cycles ?

d) the number of instruction fetches and indirect word fetches

e) the number of references directed to "tagged" pages (providing a bit in the PTW is available for designating a "tagged" page)

If only two counters are to be used, then it should be noted than an accumulation of the total number of memory cycles goes hand in hand with the measures a) - e). For that matter some combinations of a) - e) are complementary to each other in that they imply other relevant measures. Thus, there is a strong argument for more than two counters - preferrably three or more counters are desirable.

This on-line statistical data gathering could take place through an extension of the present Timer Register in the GE-645 Processor. The present Timer is 24-bits long and, therefore, can count up to 16,777,216 memory accesses. Theoretically, all the parameters to be measured, inasmuch as they are fractions of the total number of memory accesses, can be counted by 24-bit counters. However, there may be up to 12, or more, such parameters and a considerable amount of hardware will be required. Also, the Store Timer Register (STT) instruction will have to be changed to store four double words. A reasonable comprise is to have three 24-bit counters, or four 18-bit counters and extend the STT operation to one double word (72 bits). One parameter, i.e., total or programmed memory accesses, will always be measured. The other two, or three, parameters will be selected by a jumper board. The Load Timer Register (LDT) instruction will preset the Interval Timer as it does now and, in addition, will reset all the auxiliary counters. The Store Registers (SREG) will remain unchanged, i.e., the auxiliary counters will not be stored.

The measurement of the number of page table and descriptor segment references, or of the proportion of memory cycles of a given type is not expected to demand special logic since this information already exists in the Processor. The measurement of the waiting time for memory response does require additional circuitry. This is a measurement of time in seconds rather than number of cycles. The time measured should be that above the nominal response time ($INT to $PIN), which is about 700-800 nsec. A pulse trail of about 10 mc can be used for the measurement. The counter itself, however, should use increments of 1.6 or 3.2 μsec, since the average waiting time will be more than 100 nsec per access. The measurement of the distribution of transfer loop lengths requires additional hardware and may be better handled by software. The measurement of the number of references made to "tagged" pages should not present any difficulty since, to our understanding, bits are still available in the PTW. but used by Daley?

6. SOFTWARE CHARACTERISTICS

6.1 Total Segment References

In Ref. 2 a "subprocess" is characterized by the total number of segment references required to complete its execution. For a subprocess, Q_i, this statistic is denoted m_i.

It will otten be possible to obtain m_i for selected subprocesses by off-line inspection or analysis of the software. However, there will be a residue of cases where on-line measurement will be required.

There are two problems involved:-

(a) Delimiting the subprocess on the on-line system.

(b) Measuring the number of segment references from start to finish.

The first problem can usually be solved by use of the special Call, Save and Return described in Ref. 1. Otherwise a page tagging technique might be employed. The second problem is to count segment references. Clearly the additional hardware counter described in Section 4 which counted segment references directly, and which could be stored under program control would solve this problem in the most elegant fashion. However, using the processor performance measurements which give the conversion factor between segment references and time, the existing timer register could be used to obtain this statistic indirectly.

6.2 THE REFERENCE DENSITIES

The proportion of times that some subprocess, Q_i, say, references an associated paging unit, U_j, say, is denoted h_{ij}. The problems of measuring some particular h_{ij} are first to define the paging unit, and secondly to count references to it. If the paging unit is referenced during more than one subprocess it will also be necessary to establish which references were made during the subprocess, Q_i, under investigation.

It is recommended that the paging unit is defined by tagging all the correspoinding PTW's in the segment or segments which comprise it. This tag may be either a new Directed Fault or a signal to the hardware monitoring facilities as described below.

To obtain h_{ij} by hardware monitoring, an additional counter is required which would be incremented whenever a suitably tagged PTW was referenced. This could take place even though the PTW was retrieved from Associative Memory rather than Core Memory.

The other scheme, the Directed Fault technique, could be carried out roughly as follows for a tagged data paging unit. When any tagged PTW was referenced during an operand fetch, a directed fault would occur to part of the Monitor package. The Monitor routine would update its statistics, and then manufacture a copy of the PTW without the fault bit and place it in a special segment which is referenced only by the Monitor Routine. Monitor would then modify the original instruction in such a way as to reference the required page in the dummy segment, and then give an RCU to continue processing.

another one!

shared data?

There seem to be some problems in using this technique to intercept instruction fetches as well as operand fetches, and this aspect is now under study.

The hardware technique obviously only allows statistic gathering on as many P.U's as there are special counters available, while the software system is much more flexible. On the other hand, running selected portions of the software in interpretive mode might slow down some system software modules to a point at which overall system performance was unacceptable. For this reason, there is a strong case for having the additional hardware counter or counters available, even if the alternative method is sometimes adopted.

6.3 The Page Reference Distribution Function (P.R.D.F.)

As described in Reference 2 and 3, the P. R. D. F., $F_{ij}(x, p_j)$ defines the pattern in which a subprocess, Q_i, references an associated paging Unit, U_j. By implication, if there are q_j pages of U_j in core, $1-F_{ij}(q_j, p_j)$ gives the probability of a missing page fault when Q_i references U_j.

As shown in Reference 4, the P.R.D.F. can sometimes be obtained by direct computation, knowing the structure of the software. This method will be most useful in the case of Multics data bases, where the search or referencing procedure is well defined, and where the use of an interpretive trace routine might significantly affect system performance. Otherwise on-line measurement is required.

The problems of measuring the P.R.D.F. are first to define the paging unit or units of interest, and then to find the P.R.D.F. If the paging unit is referenced by more than one subprocess, it will be necessary to distinguish references by the subprocess, Q_i, of interest using the special form of Call, Save, and Return described in Reference 1.

As in 6.2, there are two methods of making the measurement,

a) Using a special counter which is incremented whenever a tagged page is referenced
b) Using a directed fault to the monitor package

Using the first technique the only measurement which can be made is the number of segment references to the selected paging unit before a missing page fault occurs. Since the Monitor package can also keep track of q_j the number of pages of the selected P.U. in core at any time, it would be possible to build up the function $F_{ij}(x, p_j)$ over a period of time.

The alternative interpretive trace technique resembles closely the method proposed in 6.2. Once again the pages of the selected paging units or units would be tagged with a special directed fault to the Monitor. After statistic gathering, Monitor would manufacture a copy of the tagged PTW in a dummy segment and carry out an R.C.U. order with the stored instruction modified to reference the dummy segment.

Statistic gathering can be carried out as follows:

If the page numbers (or page number and segment numbers, when the P. U. spans several segments) of the most recently referenced page in the P.U. is stored, then (after each reference to the P.U.) the Trace Routine can compare this number with the page number of the newly referenced page. Suppose that out of a total of n references to the P.U., there is a reference to the most recently referenced page on m occasions. Then, from the definition, $F(1,p) = m/n$.

More generally, suppose that a list of the most recently referenced pages is maintained, where x is any number less than or equal to the number of pages of theP.U. currently in core. Then the proportion of references to this set of x pages gives $F(x,p)$. The set of x most recently references pages must be updated by the Trace Routine wherever a page is referenced which does not belong to the set. Clearly the computing required to obtain $F(x,p)$ for all $x \leq p$, where $p =$ the total number of pages in the P. U., is considerable, However, it should usually be sufficient to measure $F(1,p)$, $F(2,p)$ and say, $F(p/2,p)$; $F(p, p)$ is unity by definition. Then $F(x,p)$, which is a smooth monotonically increasing function of x, can be obtained by interpolation.

As in 6.2, the hardware alternative is much faster to run. At the same time statistic gathering may be very inefficient since Monitor has no control over the number of pages of the selected paging unit present in core at any time. By contrast, the software solution can gather statistics on F_{ij} (x, p_j) for any selected value of x, provided there are more than x pages of U_j in core for a significant processing time.

On balance, the software solution is preferred. However, the additional hardware required is identical with that necessary to monitor the reference densities h_{ij} (Section 6.2), and hence there is a good case to use the hardware solution when the use of an interpretive trace would prejudice the overall performance of the system.

REFERENCES

1. "Software Tools for Monitoring and Tracing in Multics"; J. F. Gimpel , Multics B0039; February 15, 1966

2. "Proposal for Stage 1 Simulation of the GE 645 - Multics System;" G. A. Shippey; February 15, 1966; M & A Report 24/1b

3. "Simulation of I/O and Secondary Storage in a Multics File System Model"; D. C. Zatyko, February 18, 1966; M & A Report 25/1b

4. "A Proposal for Modeling Associative Memory Performance"; J. E. Shemer, February 3, 1966; M & A Report 19/1b

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6/12 1966

Memo to Saltzer Room Ext.

1. Not clear Instruction mix includes indirection
2. p.5, They should be told of Gimpel's non-role. They still have a problem to solve.
3. Perhaps they should have a man join us and work with us for a while?
4. I prefer the 24-bit case as long as there are jumpers
5. Extensibility in PTW & conflict w/ new prot. ideas?

from Corby Room Ext.

MURAN BOSTON

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Memo to Corby ... 5/23 19... Room ... Ext. ...

Comments?

I have a copy too.
I haven't had time to read
it yet but I intend to.
6/12 Corby

from Jerry ... Room ... Ext. ...

MURAN BOSTON