THE ANALYSIS OF COMPUTER SYSTEMS FOR PERFORMANCE OPTIMISATION

PIERRE ANDRé MEIRING

Submitted in partial fulfilment of
the requirements for the degree of
M.Sc.Eng.

Department of Electronic Engineering
University of Natal

PRETORIA
1987

I wish to certify that the work reported in this thesis is my own original and unaided work except where specific acknowledgement is made.

Signed [illegible]

P. A. MEIRING

Acknowledgements

The author wishes to record his sincere appreciation for the assistance given by the following :

Professor H.L. Nattrass, Head of the Department of Electronic Engineering, University of Natal, who supervised the work.

The CSIR for sponsoring the project through a Post Graduate Study Bursary.

Messrs D.C. Levy and R.C.S. Peplow of the Department of Electronic Engineering, University of Natal, for their help and technical advice.

Messrs D. Everett and D. Beard for their aid in building and installing the electronic equipment developed and used in the project.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1.	B	Cyclic Data Transfer Rate of I/O devices
2.	Bs	Transfer Rate between External and Main Memory
3.	COMPAS	Computer Performance Analysis System
4.	D	Data Path Width of I/O devices
5.	GE	System Global Elements
6.	G_p	Nominal Capability of System Resource
7.	G_{pp}	Measured Utilisation of System Resource
8.	IAG	Information Acquisition and Dissemination Global
9.	IAP	Information Acquisition and Dissemination Particular
10.	If	Importance factor of SCP
11.	IPG	Information Processing Global
12.	IPP	Information Processing Particular
13.	ISG	Information Storage Global
14.	ISP	Information Storage Particular
15.	It	System Idle time
16.	I_y	Workload Need Factor
17.	K_c	SCP Workload Constant
18.	K_i	IAP Workload Constant
19.	K_p	IPP Workload Constant
20.	K_m	ISP Workload Constant
21.	Lf	Loading factor of SCP
22.	Nb	Number of Bits in External Memory

23.	Nw	Number of Words in Main Memory
24.	P	Workload Type Priority
25.	PE	System Particular Elements
26.	PSYS	System Performance
27.	SCG	System Control and Capability Global
28.	SCP	System Control and Capability Particular
29.	Tf	Memory Time Factor
30.	Tr	Data Transfer Rate of I/O devices
31	U	Workload Type Usage Factor
32.	Uf	Usage factor of SCP
33.	Ut	Unit time
34.	Ws	Word size in bits

ABSTRACT

The project investigated the problem of performance optimisation of computer systems at the systems level. It was ascertained that no generally accepted technique for approaching this problem exists. A theoretical approach was thus developed which describes the system, the workload and the performance in terms of matrices which are deduced from measured data. An attempt is then made to verify this theory by applying it to a real system in a controlled environment. A dummy workload is used and measurements are made on the computer system for various configurations. The results thus obtained are compared with the expected trends in system performance and conclusions are drawn which appear to verify the validity of the theory proposed.

1. INTRODUCTION

The world is experiencing a computer boom of staggering dimensions. Systems are getting larger and more complex by the day and the users are coming from a wider spectrum of the population. The days when computers were run by specialist personnel on a batch run basis are gone forever, and amidst all this one very significant factor has emerged, the user has an insatiable desire for an improved service. Interactive operation is no longer a privilege reserved for only a few, and speedy responses to even the most complex workloads are demanded.

A study of the computer performance analysis discipline revealed that it is the poor relation in the computer systems' engineering field. Although there is a great deal of work done on improving computers and the results achieved are astounding, at the system level one finds a lack of precise knowledge, due to the extreme complexity of the field. It was found that various investigators all perceived similar problems, namely the lack of a formal definition of system performance and a lack of agreement on techniques to be applied. Furthermore, although various aspects of system performance could be analyzed in detail, each of these techniques was applied in isolation from the rest of the system and thus disregarded parasitic effects and mutual incompatibilities and were often based on complex mathematical models or theoretical workloads. The result was that most techniques tended to have serious inadequacies when confronted with real world systems.

It was thus decided to carry out an investigation in the field and attempt to develop a performance analysis methodology, basing the technique on simple premises and avoiding complex mathematics. As described in Chapter 3 the first phase consisted of a study of the system structure to identify the components of the system and define the elements and bounds of this system in such a way that they are usable. The result was a matrix equation which defines each element of the system in terms of its capability to provide a usable resource to the environment. The environment, or workload, is defined in terms of the importance which it attaches to each of these resources, not in terms of a theoretical model, but in terms of measurements made on the system.

The result is a complete system definition which has the advantage of being generalized and is based on the real world computer system on which it is to be used.

Having developed the skeleton of the proposed technique a more detailed analysis was done of the individual elements of the system and the methodology to be applied to converting the theory into practical results. Four main elements, to be known as the System Global Elements, which describe the basic structure of any operational computer system were identified.

The Information Processing Global, or IPG, describes the system's capability to manipulate or process the information stored in the computer and lies at the heart of its performance capability. The Information Storage Global, or ISG, covers the total spectrum of subsystems which can be used to store programs and data within the computer, while the Information Acquisition and Dissemination Global, IAG, describes the mechanism by which programs and data are input to the system for storage and use, or output from the system for external use or interpretation. The above three globals describe the physical hardware making up the system.

The fourth main element, the System Control and Capability Global, SCG, describes a concept and not a physical entity and deals with the manner in which the system's physical elements co-exist and operate. It also describes those tasks carried out by the system for the user and is in a broad sense analogous to the Software Operating System,SOS.

Armed with a detailed theoretical analysis the project now progressed to the stage of illustrating the validity of the technique on a specific system. The quantities which had to be measured were determined and the techniques to do so were developed. The result was a system specific set of tools which, however, applied general principles that could be used on virtually any system. These tools consisted of software capable of extracting data from the operating system tables to determine the nature of the workload and the loading on the system memory and a hardware analyzer the COMPAS, or Computer Performance Analysis System, with comparators, counters and data storage facilities which could monitor the target system's busses and extract data concerning processor activity, memory usage, input/output activity and usage of the external memory or discs in the system. The COMPAS system is described in Appendix A.

Measurement results were presented as arrays of data which, after suitable reduction using a suite of programs, gave the analyst a variety of information. Presentation was in the form of histograms of system activity versus memory address, used to analyze the operation of programs, traces of activity within a program and graphs of resource utilization versus time.

This data was then used to ascertain the validity of the techniques proposed and provided an accurate and detailed picture of the system operation. A series of tests were conducted using a dummy workload and the effects of changes to the system were then ascertained, while the validity of the integrated model was demonstrated.

It must be emphasized that a full scale analysis of an operational system was not attempted due to the experimental nature of the techniques. A basis was, however, created for future investigations and the validity of the approach to performance analysis was confirmed.

2. BACKGROUND AND OBJECTIVES

During the last decades computer engineering has advanced in great strides and today highly complex computing systems exist. There can be no doubt that these systems perform the functions that they are designed for correctly. However, being engineering products, it is also true that they must meet certain other performance specifications before being acceptable to the user community. It is for this reason that the performance measurement, analysis and optimization of computer systems is of interest.

Much work has been done on the improvement of computer system components. New programming techniques have been developed, while the hardware in particular is constantly being evaluated and improved. This work is all aimed at the improvement in performance of individual subsystems and Myers (1982) quotes this as the reason for the present day preoccupation with "addition and multiplication times" as measures of performance. Bell and Thorley (1985) agree with this assessment and describe it as a " project oriented approach " where " performance analysis is undertaken to attack specific problems ".

On the other hand, the analysis of complete systems as an entity is a relatively undeveloped field in the computer engineering environment. Its purpose is to evaluate the operational effectiveness of the entire hardware-firmware-software combination rather than the individual elements of the system. This aspect of computer system evaluation remains the poor relation, according to Kuck (1978), due to the fact that it is an imprecise and highly complex field. Indeed, so difficult is it to characterize computer <u>system</u> performance that at present no generally acceptable definition exists.

According to Ferrani (1978) the major problems in the field are the lack of quantitative laws to form the foundation of scientific studies and the fact that this aspect of computer engineering is still regarded as being an art and is taught on a qualitative and descriptive basis in the vast majority of curricula dealing with computer organization.

In addressing the problem in this study an attempt is made to develop a workable and general definition for system performance based on simple logical premises. The approach followed is not based on anything found in the literature and is in fact at variance with the thoughts of authors in the field. Instead the aim is to take a global view of the problem, beginning from basic principles, and work to a complete solution of the problem. This definition can then be used along with practical evaluation techniques in the analysis and optimization of the performance of computer systems.

Definition of the problem forms the basis of any analytic study. In this case two things must be looked at. Firstly the definition of computer systems and secondly the definition of performance as it relates to these systems. Figure 1 shows that computer systems can be described as closed loop systems. Users generate the workload which is converted into performance by the computer installation. This performance then influences the user community and thus the workload.

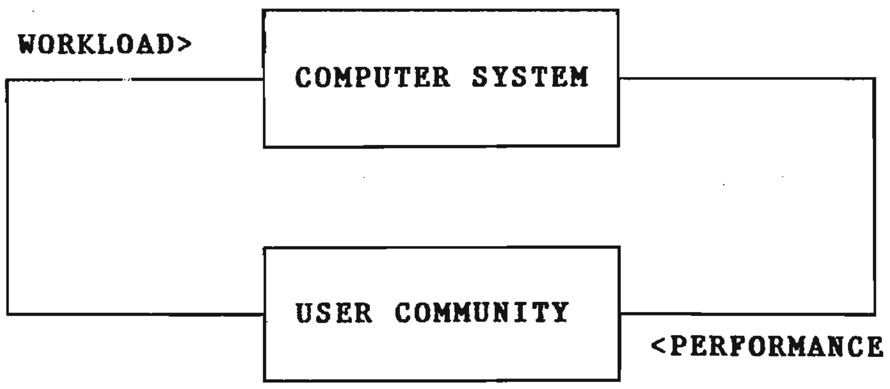


Figure 1 Block diagram of a computer system

Unfortunately the definition of performance is a very subjective topic. Different evaluators regard different aspects of performance as being important. Various authors in the field discuss the problem. All agree that many different measures of performance exist but there is little evidence of agreement as to the best choice.

Kobayashi (1978) divides the measures into two classes, the user oriented measures such as terminal response time and the system oriented measures such as job throughput. Svoboda (1976) lists sixteen measures of performance and Ferrani (1978) a similar number. Table 1 shows a number of examples of such measures.

Table 1 Main classes of quantitative indices of computer system performance

Index class	Examples of indices	General definition
Productivity	Throughput rate Production rate Capacity(maximum throughput rate) Instruction execution rate Data processing rate	The volume of information processed by the system in unit time
Responsiveness	Response time Turnaround time Reaction time	The time duration between the presentation of an input to the system and the receipt of the appropriate response to said input
Utilization	Hardware module (CPU, memory, I/O channel, I/O device) utilization Public software module (e.g. compiler) utilization Data base utilization	The ratio between the time a specified part of the system is used (or used for some specified purpose) during a given interval of time and the duration of that interval

This does not help a great deal as the question still arises as to which measure is to be used and this seems to depend on who is to use it. The ideal would seem to be to express performance in such a way that it indicates all aspects of performance in an objective way. The specification of what constitutes the system is also subject to many interpretations and here again a rigid definition is desirable. The approach to be used here is to translate all these many facets of computer performance and structure into a set of mathematical equations.

Looking at the system itself certain problems arise. The major of these is that the system by its very nature interacts with its immediate environment. This environment contains human users impossible to describe in technical terms. The system itself can also contain human elements if it requires human intervention to operate correctly. To overcome this problem it is proposed that the evaluation technique be based on measured data. This obviates the necessity for prediction of actions and thus solves the problem.

A further problem arising during the system description phase is the great variety of systems which exist. According to Lorin (1982) one of the greatest challenges facing system designers and users is to find a common denominator amongst the various configurations and architectures found in computer systems to allow them to develop a common strategy for dealing with these systems. Their structure and capability can vary so greatly that a common definition seems impossible.

When dealing with installations ranging from pocket calculators to systems such as the Control Data Cybernet system which spans several continents, it is not the peculiarities but rather the common elements which become important. These common elements once identified lead naturally to the system description proposed and allow any depth of detail to be specified as shown in the next chapter.

From the point of view of performance many of the same problems apply. The environment in which the system operates is of great importance as this is what determines both the loading on the system as well as the aspects of the system which are important. For this reason description of the environment is vital to any performance evaluation. In this project a description technique based on measurement of the actual workload, this being the physical manifestation of the environment on the system, is employed. Parameters describing the workload, derived from the measurements, are used directly in evaluating the performance of the system.

Looking at the performance of the computer system elements themselves, it is found that the individual elements within the system can be evaluated. The problem as stated by Hayes (1978) is that no single one, or group of elements, is capable of expressing the performance of the entire system in all situations. Furthermore, when attempting to optimize them individually it is soon apparent that many of them are mutually exclusive. Examples are processing throughput and system capability; the more processing time required by the workload the less available to the system software, thus limiting the complexity and capability of this software.

This is the kernel of the problem, since different user requirements need optimization of different system characteristics. No hard and fast definition of system performance describing all systems can thus be made, but if the opinion of Croft and Cantrell (1986), who describe a computer system as a structure where the workload generates equipment utilisations and a corresponding user experienced performance, is accepted then in general it can be said that <u>an optimized system</u> is one which will complete a specified set of tasks or workload with the least utilisation of resources.

This definition of an optimum system contains two words which hold the key to the entire problem, they are "resources" and "tasks". To begin with the system is defined as the sum of its resources, the environment as the sum of the tasks, the performance as the ability of the system to perform these tasks using the resources within the system. Obviously if the workload is regarded as being a constant then the smaller the fraction of the total resources used by the workload the higher the performance of the system.

This approach to computer system performance is at variance with the opinion of many authors in the field. Ferrani (1978), for instance, holds that performance evaluation cannot be system and application independent. However, looking at the very latest simulation and modelling technique queueing network analysis, as described by Lazowska et al (1985) and Mac Nair and Sauer (1985) the basic cornerstone of the technique is its capability to describe an entire system with reasonable accuracy using a set of general building blocks. This results in a model of a system which inherently descrbes both the workload and the system structure.

In the same way the techniques described here are designed to be system and workload independent. This is achieved by defining a similar set of building blocks which are capable of implicitly expressing the nature of the system and the workload in the equations which describe the performance.

The objective is thus to identify the resources of computer systems in common terms, prioritise them based on the measured workload and then determine the performance of the system in terms of the loading of these resources.

3. THE EQUATION DESCRIBING PERFORMANCE

3.1 INTRODUCTION

The equation is developed in three stages. Stage one deals with the physical attributes of the system, its structure and the nominal performance capability of each element or resource within the system. Next the workload is introduced and its effects on the system are incorporated into the equation. Finally the actual performance of the system elements is introduced.

Once the basic equation has been described each global element is discussed in detail and the various system parameters that need to be measured in order to determine system performance are ascertained. Knowing what to measure it is then possible to proceed with the development of the necessary tools for gathering data on a real system in order to verify the validity of the theory.

3.2 THE GENERAL SYSTEM DESCRIPTION

This section has as its aim the development of a set of general equations which may be used to describe the total resources of a computer system of any kind. To begin with the types of resource which make up a computer system are identified. These Global Elements, GE, are then expanded to be capable of expressing the detail of each element so that even the smallest difference between two systems may be clearly illustrated. This expansion of the GE leads to the System Particular Elements, PE, each of which describes a particular detail of the system structure.

Taking a simple micro-processor system as a Computer System Model, certain elements common to all systems become apparent and they are the Globals. There are four globals identifiable in all systems, three are physical entities and the other one conceptual. This conceptual Global is nevertheless central to the existence and operation of any computer system.

The ability of a computer to process data, or compute, is the first GE since this is the central concept on which computers rest. This is the Information Processing Global, IPG, and from this follows the second GE, namely the Information Storage Global, ISG, since the computer must store the data which is to be processed as well as the program which tells it how to process the data. The third GE is the Information Acquisition and Dissemination Global, IAG, which relates to the way in which the computer acquires the data and programs from its environment and disseminates the results of its processing back to the environment. This covers the three physical types of GE mentioned.

Each of these GE can conceivably consist of several different physical resources such as multiple parallel processors, different types of memory and a vast array of peripheral interfaces or equipment such as printers, VDU's and keyboards. In order to describe each of these elements the PE are defined. The IPP or Information Processing Particular Elements describe individual processors within a system, the ISP or Information Storage Particular Elements describe individual types and areas of memory within the system whilst the IAP or Information Acquisition and Dissemination Particular Elements describe individual peripheral devices or interfaces.

The fourth GE, the SCG, is conceptual. It is, however, no less important than the others, since it enhances the capabilities of the physical resources by making them useful to the computer environment. This global thus encompasses the software operating system and all its elements such as device driver routines, program schedulers, memory management software and a variety of other software which eases the users' task in using the system. Each of these routines or programs then make up the SCP or System Control and Capability Particulars.

Each of the individual resources within the system is now identified and it remains to arrange them in such a way that the system description can be manipulated mathematically. To do this the individual elements are arranged into a matrix format with four rows and N elements where each element is related to a PE in the form shown below.

$$\text{SYSTEM} = \begin{bmatrix} IPG \\ ISG \\ IAG \\ SCG \end{bmatrix} = \begin{bmatrix} IPP_1 & IPP_2 & \ldots\ldots\ldots & IPP_n \\ ISP_1 & ISP_2 & \ldots\ldots\ldots & IPS_m \\ IAP_1 & IAP_2 & \ldots\ldots\ldots & IAP_o \\ SCP_1 & SCP_2 & \ldots\ldots\ldots & SCP_p \end{bmatrix}$$

where n,m,o,p are equal to the relevant number of PE for the global and all vacant locations would be set to zero.

This matrix describes the system structure and its elements, but is insufficient to describe the performance of a computer system. The reason for this is that it does not include the effects of the environment. These effects or constraints on the system are important since they determine the importance of the various aspects of the system in operation. To put it another way, it is true to say that the operating environment of a computer system determines those aspects of a computer which are important and since the physical manifestation of this environment encountered by the computer is in the form of programs which are run on the computer, it is these programs and the resource load they place on the computer which is important.

This is in effect the workload placed on the computer.

3.3 THE WORKLOAD

In general two approaches are followed when the problem of describing the workload is addressed. Most authors in the field make use of hypothetical models based on mathematical probabilities of instructions occurring. This method leads to techniques such as the use of instruction mixes. Certain other authors, however, reject this approach for two reasons, namely the fact that they are aimed mainly at the processors' capabilities and do not exercise the other resources in the system and secondly they tend to be based on complex mathematical models which cannot be verified very easily. This viewpoint is supported by Bard (1976) who did extensive studies on the IBM VM/370 operating system. His findings were that a workload model based on measured data generally gives results which are the closest to the actual workload on a system at any time. In this study this principle is carried even further since the workload is based on measured data at all times.

It has been established that the workload is the physical manifestation of the outside world on the system. In order to assess the performance of the system it is necessary to know what is required of it and to this end it is necessary to develop a technique which describes this workload and relates the workload needs to the system resources.

Looking at the system description it is true to say that it can describe the system structure to any depth required. Each element described is one of the resources of the system and the workload according to its nature places differing demands on these resources. By quantifying these demands and relating them to the individual resources it is possible to fully describe the workload and the performance demands made on the system by it. This is the essence of the workload description technique proposed here.

As with the general definition of computer systems there are a large number of workload definitions extant, and once again the model proposed is an attempt at producing a generally acceptable and simple model based on a practical approach to the problem.

Looking at the workload concept on a macroscopic level it is true to say that the workload is defined to be : "The execution of all those duties and tasks required of the system by the user community" where the user community is itself defined as "any person or object providing the system with a stimulus to which it reacts, and from which stimulus a resulting stimulus is output beyond the bounds of the system".

The bounds of the system are simply defined as the limits of the system as described by the system structure globals previously defined.

The workload is thus made up of numerous tasks which the system must perform. On one system these tasks may be easy to define and describe, an example being the functions on the keyboard of a calculator. For others such as an industrial process control system, it would be far less simple. When dealing with complex systems the first step is to divide the tasks into groups or types, the larger the number of types the more detailed the workload description and since each task consists of a set of input stimuli which cause system reactions resulting in a set of output stimuli, it follows that the groupings be made on the basis of these actions and stimuli. It also follows that if the stimuli and actions are similar, and because system actions are based on programs, groupings can be made on the basis of the programs run on the system.

Definition of the system workload thus consists finally of ascertaining all programs which run on the system. Once this is completed it is necessary to decide on groupings. In some systems where the number of programs is small it is possible to define each program as a type of workload, in others the grouping may be less simple and must be based on similarity.

This similarity must be expressed in terms of program resource requirements. Any system contains certain resources based on its physical configuration. These resources correspond to the elements of the system. Thus, for example, programs which use similar percentages of processing time, have a similar number of input/output actions to the same devices, use the same system capabilities and so on, in unit time, are regarded as being similar, even though they may in fact be doing completely different tasks. A resource requirement table must thus be drawn up for each program and in this way the programs can then be grouped. This resource requirement table has a second use, however, as is discussed below.

The whole object of the workload definition is to facilitate the assigning of factors of importance to the elements of the system. This is done by assigning to each resource in the system a need factor corresponding to the usage by each program. Thus in the case of a program which does a lot of processing but little else, the largest need factor would be applied to the processing resource and only small need factors, if any, to other resources. This need factor corresponds to the values placed in the resource requirement table drawn up for each workload type.

Next the usage of each workload type is discussed, that is to say how frequently is a type of program run on the system. Lastly the importance of each type of workload to the outside environment, that is its priority, is covered. Combining these three factors, namely resource requirements, usage and priority for each type of workload gives a complete description of the various types of workload. This results in four workload constants Kp,Ks,Ki and Kc which are each coupled to the four Global variables IPG,ISG,IAG and SCG. These constants can also be further sub-divided to relate to the PE. Combining the various types then results in a complete workload description.

Thus for each resource in the system a workload constant, which is the sum of the workload constants for the various types of workload, is found. Each type constant is in turn the product of the usage, the need factor and the priority.

Expressed mathematically the workload is thus :

$$W = \begin{bmatrix} Kp_1 & Kp_2 & \ldots\ldots & Kp_n \\ Ks_1 & Ks_2 & \ldots\ldots & Ks_m \\ Ki_1 & Ki_2 & \ldots\ldots & Ki_o \\ Kc_1 & Kc_2 & \ldots\ldots & Kc_p \end{bmatrix}$$

where Ky_i is the i'th workload constant of global y and is expressed as

$$Ky_i = Ky_{i1} + Ky_{i2} \;\ldots\ldots + Ky_{ig}$$

for g types of workload, and

$$Ky_{iq} = P_q \times U_q \times Iy_{iq}$$

where P_q is the priority factor of workload type q, U_q is the usage factor and Iy_{iq} is the need factor for particular element i of global y.

This matrix fully describes the workload on the system and has a major advantage over conventional models in that it takes into account the priority of the individual workload types.

3.3.1 Characterization of Workload Resource Requirements

Characterizing the workload consists of determining the way in which each part of the workload contributes to the total loading of the system and then of determining the resource needs of each individual workload type.

The individual workload types are discussed in terms of their relative importance to determine the value of

P the priority factor. Most computer systems handling large volumes of work have some form of priority structure which accords precedence to one program over another. In this case the priority structure leads naturally to the value of P since it is already stated. To incorporate P into the workload constant it need only be normalized. In cases where there is no priority structure one of two courses can typically be followed, the P factor can be discarded in which case all workload types are accorded equal priority or the evaluator can arbitrarily assign values to workload types on the basis of his knowledge of the system or a survey of the users' needs.

The second task is to determine the usage of individual programs. This is done by designing a tool which measures the frequency with which a particular workload type is the scheduled program for execution on the system. This can be done using a hardware or software monitor on the system which interrogates the system at a frequency giving a time interval shorter than the shortest run time of any workload type. The frequency of occurrence of each workload type must be accumulated and over a period of time an accurate picture of the workload distribution on the system can be assembled. The only restraint here is that if a software monitor is used, it must not load the target system significantly or the recorded data will be disturbed. This process gives the usage factor U.

Finally the need factor for each resource, Iy_i must be determined. This is the significance of each resource within the system to the workload type, relative to all others. To illustrate this two programs, which have a similar amount of input/output in unit time, are used as examples. Let the one, however, be a simple editing type program where data is input, stored in memory and output, whilst the other program inputs the data, does some very complex processing of the data, updates a large database and outputs the new data. Clearly the two programs have similar input/output resource requirements but in the case of the first program, this input/output requirement is the primary requirement, whilst for the second program mentioned the requirement could very easily be far less important than say, sufficient processing time or the availability of large external memory.

What is thus required is some method of assigning the importance factor, which takes into account the structure and operation of the program. The method used in this study is based upon the fact that allied to any action in the system there must be CPU action, thus based on the percentage of processing done to complete any action the importance of that action is determined.

The processing need factor is assigned a start value of 1. All other need factors are now determined by dividing the processing needs amongst these resource requirements, based on the percentage of processing time required. Each of these need factors will then be less than one.

The various factors are determined using a mixed listing of the program, a load map of the program and a histogram of the program operation.

The only aspect of the program not covered by this technique is the storage requirement in main memory. Here the need factor is calculated by dividing the available memory into the requirement for the particular program, thus a program requiring the entire memory has a value of 1.

Having determined the need factor for each resource the values are normalized to give the relative need factors for each resource within a particular workload type.

Having defined the workload the concept is carried a step further.

3.3.2 The Multiple Workload Concept

In many systems the workload can vary dynamically due, for instance, to the time of month. An example is the high loading on systems preparing end of month accounts or end of year tax returns. In other cases it is possible to recognize certain distinct sub-workloads on the system. It is thus necessary to be able to describe the system taking into account only these parts of the workload. In this case the weighting factors used are simply those attributable to that part of the workload or that sub-workload.

Thus each global weighting factor can be thought of as the sum of the factors contributed by each part of the workload or each sub-workload.

Thus :

$$Kx = Kx_1 + \dots + Kx_n + \dots + Kx_h$$

where Kx_n is the weighting factor due to subload n of Kx, which is the weighting for Global x.

This now leads to the complete workload description which is the sum of all the sub-workloads expressed mathematically as :

$$W = \sum_{a=1}^{n} \left(\sum_{b=1}^{g} Kp_{ab} \right) + \sum_{a=1}^{m} \left(\sum_{b=1}^{g} Ks_{ab} \right) + \sum_{a=1}^{o} \left(\sum_{b=1}^{g} Ki_{ab} \right)$$

$$+ \sum_{a=1}^{p} \left(\sum_{b=1}^{g} Kc_{ab} \right)$$

for a system with g types of workload.

The resulting matrix as originally described on page 18 thus has constant elements which would individually be described as below :

$$Ks_1 = Ks_{11} + Ks_{12} + \ldots\ldots + Ks_{1g}$$

Each element is the description of the importance placed on each PE in the system by each workload type.

The system and its environment can now be described completely. The various parameters describe the system resources whilst the associated constants describe the environment. Thus any system can be described as follows :

$$SYS = \begin{bmatrix} Kp_1.IPP_1 & Kp_2.IPP_2 & \ldots\ldots & Kp_n.IPP_n \\ Ks_1.ISP_1 & Ks_2.ISP_2 & \ldots\ldots & Ks_m.ISP_m \\ Ki_1.IAP_1 & Ki_2.IAP_2 & \ldots\ldots & Ki_o.IAP_o \\ Kc_1.SCP_1 & Kc_2.SCP_2 & \ldots\ldots & Kc_p.SCP_p \end{bmatrix}$$

This is a somewhat unwieldy looking description, but only rarely will it be necessary to break the system down into its components whilst n, m, o, and p will be small for most systems. Generally the expression is far more manageable than it would seem to be at first glance. The beauty of the system is that it is in fact possible to isolate and describe even the smallest aspect of the system whilst still taking into account all other aspects of the system in a general way. This allows great flexibility in the examination of any computer system.

3.4 THE DEFINITION OF PERFORMANCE

The definition of computer systems given contains within it a complete description of the system structure and its environment. It is also true that each element or resource has a finite capability. For instance the memory of a computer has a fixed size. Furthermore from the original definition of performance it is known that the less loading which a specific workload places on these resources the higher the performance. Thus by determining the performance of each individual element in the system structure and integrating this performance with the system equation in place of the resource description, we will have an equation which describes the system's performance for the given workload.

The performance of each individual element is determined by taking the nominal capability of the element, which is known from the system specification and measuring the actual loading of the same element during system operation. From this data the spare capacity is then determined and expressed as a fraction of the nominal capacity. This is then the performance which is expressed mathematically.

$$\text{Performance} = (G_p - G_{pp})/G_p$$

where G_p is the nominal capability and G_{pp} is the measured load on the resource in question.

This then leads to the definition of each particular element as shown in the example below.

$$IAP_2 = (G_2 - G_{2p})/G_2$$

where G_2 is the nominal capability of information acquisition and dissemination resource two and G_{2p} is the measured loading of the resource.

Determining the values for each particular element and plugging these values into the matrix description of the system gives a complete description of the system performance. Adding the individual rows of the matrix gives the performance of the particular global and adding all the elements in the system gives the total system performance. This performance value has a maximum of one since the workload constants for the entire system are normalized and add up to one whilst the maximum value of performance for each element is one.

Expressed mathematically system performance is thus :

$$\begin{aligned} PSYS &= \sum_{a=1}^{n} (Kp_a.IPP_a) + \sum_{a=1}^{m} (Ks_a.ISP_a) \\ &+ \sum_{a=1}^{o} (Ki_a\ IAP_a) + \sum_{a=1}^{p} (Kc_a\ SCP_a) \le 1 \end{aligned}$$

This then gives the performance of the system under a particular workload. If the system is now optimized the improvement is shown as an increase in the spare capacity for the load in question and the value of PSYS thus increases. The maximum value is now reached when each resource is totally unloaded for a particular workload or in other words the workload in question places no visible load on the system.

Expressing the performance in this way has the advantage that it is immediately apparent which resources in the system are reaching maximum capacity. This is important since these bottlenecks can seriously degrade performance.

3.5 PERFORMANCE OF THE SYSTEM ELEMENTS

This section deals with the detailed analysis of the technique employed to determine the structure and performance of each particular element in the system. Each type of global element is dealt with separately.

3.5.1 The Information Processing Global

This Global covers all information manipulation done by the system. In the simplest system there is only one processor. It carries out all information manipulation according to predetermined rules and at fixed rates. Such a system has only one Processing Particular, the central processor.

In more complex systems there could be multiple processors with differing functions. Many processor types are identifiable and they vary greatly in capability. The most complex type exists in large single processor systems where one processor must handle memory management, input/output, data manipulation and conversion as well as system control processing. Other types are : peripheral processors, which only deal with input/output functions, memory control processors which specifically care for system memory or sections of memory, program processors which deal with information processing or, stated simply, deal with "number crunching", and control processors which exercise control over one or more of the above. The functions of these processors can be combined to form an even greater number of types.

Normally the capability of a processor is expressed in terms of the speed with which it can process a given task or benchmark or else in terms of the number of instructions it can execute in a second. The former method requires dedicated testing of the processor to implement, has been found to yield greatly differing results based on the particular benchmark to be used and does not take the workload into account in any way and for this reason can be discarded immediately in this study.

Looking at the second alternative usually used in its most simplistic form the result is usually expressed as a constant, being the average for all types of instructions, for any one machine, each type of instruction using a fixed number of basic machine cycles to execute. No notice is taken of the complexity of the processor instruction set or the length of code normally generated by different applications. Furthermore the normal instruction cycle time can in most machines be influenced by such outside factors as cycle stealing and wait states induced by slow memory or peripherals. Once again no attention has been given to the workload's influence.

An extension of the second method has, however, been developed where the instruction cycle time average is replaced by a value calculated on the basis of instruction mixes. Studies have been carried out by a number of researchers to determine the instruction usage distribution for various workloads.

Examples given in Ferrani (1978) for standard instruction mixes are given in Table 2. Instruction mixes are, however, subject to a wide variety of problems. Borovik and Neumann (1979) and Ferrani (1978) discuss the use of instruction mixes and conclude that they are often very sensitive to workload variations and the same workload on different systems often generate vastly different mixes as is obvious from the examples given.

Furthermore, using instruction mixes, still does not take into account the volume of processing produced by a specific workload.

Table 2 Two standard instruction mixes

Instruction class	Gibson mix, fi(%)	Flynn mix, fi(%)
Load/store	31.2	45.1
Index	18.0	
Branch	16.6	27.5
Compare	3.8	10.8
Fixed point	6.9	7.6
Floating point	12.2	3.2
Shift/logical	6.0	4.5
Other	5.3	1.3
	100.0	100.0

Because of the obvious inadequacies of the normal methods of expressing processor performance a third approach was decided upon. Performance of a system particular element was defined to be :

$$\text{Performance} = (G_p - G_{pp})/G_p$$

Now defining G_p to be maximum processor performance and G_{pp} to be the actual load on the system the equation becomes :

$$P = 1 - G_{pp}/G_p$$

where G_{pp}/G_p is the fraction of the processing resource actually used thus :

$$1 - G_{pp}/G_p = It/Ut$$

where It is the system idle time in unit time and Ut is the unit time.

IPP is thus directly given by the fraction of time that the system is idle and is expressed as :

$$IPP_n = It_n/Ut$$

This approach has several advantages. Firstly the use of instruction mixes and benchmarks is avoided and the method is workload independent. Secondly the method can be applied directly to an operational system without placing constraints on the workload or the user community and lastly it is generally a fairly simple exercise to measure the idle time of a processor.

3.5.2 The Information Storage Global

All systems operate under the control of a sequence of instructions, the program, on a set of data. For this reason they must contain some element within their structure where the program and data, the information, is stored. In the trivial case of calculators the program is simply a single instruction stored in the hardware configuration of the machine and the data is entered into a register of the machine. Thus the storage is simple.

In general, however, multiple instructions are sequenced to create programs and these programs as well as the data operated on, is stored in the system memory. System memory can further be divided into two classes, internal or main memory and peripheral or external memory.

Internal memory is that portion of system memory which can be accessed directly by the processor, usually by means of a single instruction in the case of data and automatically in the case of instructions. This main memory can either be reserved exclusively for programs or data or be used for both types of information; furthermore, it can be either dynamic in which case the processor can change the information stored in it or static as in the case of Read Only Memory.

External memory on the other hand is generally used as mass storage and when required the specified information is transferred to main memory for the immediate use of the processor (by processor a part or the entire system processor facility is meant).

All systems have a main memory element but the external mass memory element is optional. When attempting to express the information storage global mathematically the two types must be approached individually. In the case of main memory the standard against which the Main Memory Particular Element is measured is the maximum memory directly addressable by the processor. This can include memory controller elements which allow switching between areas of memory. Generally, however, the standard is based on the size of the address bus. Thus, in the case of a system with a twenty bit address bus, the maximum would be one mega-words. In addition this size standard would have to take into account the actual word size of the memory.

Expressed mathematically the memory size is :

$$\text{Main Max} = 2^{ab} \times Ws \text{ in bits}$$

where ab is the address bus width and Ws is the word size in bits.

This expresses the memory size but the access speed of this memory must also be taken into account. This aspect is important as it determines the number of instructions or the amount of data which the processor can use in unit time given that the processor instruction cycle time is faster than the memory access time. To express the memory capability correctly the time element is taken as the instruction cycle time divided by the access time provided that the value may not be greater than one. This time factor then multiplies the value of the memory particular element. In many systems it will be one. Thus for any particular area or type of main memory the nominal value of that Particular Element is :

$$ISP = Nw \times Ws \times Tf$$

where Nw is the number of words, Ws is the word size in bits and Tf is the time factor.

To determine the performance of ISP for main memory particular elements the maximum values of Nw, Ws and Tf and the actual values of Nw, Ws and Tf during system operation are found and plugged into the performance equation. Assuming that in most systems Tf is one and Ws is a constant then ISP is expressed as :

$$ISP_m = (Nw_m - Nw_{mp})/Nw_m$$

where Nw_m is the maximum value and Nw_{mp} is the actual amount of memory used by the workload.

In the case of external memory the actual storage elements are often interchangeable, the actual size is thus taken as the total amount of memory which can be mounted on the system at any one time. Now while this expresses the size of external memory it does not express the entire capability of this memory. This capability is also dependent on the ability of the system to access this memory. This factor is in fact more correctly a part of the information acquisition and dissemination global. For completeness sake an external memory accessibility factor which expresses the ability of the system to access the external memory is, however, included.

ISP for external memory elements is therefore defined as :

$$ISP_m = Nb \times Bs$$

where Nb is the size in bits of the external memory device and Bs is the number of bits accessible in unit time.

Assuming now that the size remains a constant, which need not necessarily be true but is often the case, then the equation for performance becomes :

$$ISP_n = (Bs_n - Bs_{np})/Bs_n$$
$$= 1 - Bs_{np}/Bs_n$$

where Bs_n is the maximum data transfer rate and Bs_{np} is the actual load generated by the workload on ISP_n.

These expressions thus define memory performance for the system.

3.5.3 The Information Acquisition and Dissemination Global

Interaction with its environment is one of the most important aspects of any computing system. This interaction can be divided roughly into two types, namely machine-machine interaction and man-machine interaction. The former covers such areas as instrument control and automatic monitoring as well as the process whereby external mass memory is accessed.

The latter covers the inclusion within the system of all terminals, printers and plotters and such devices as either input information directly from users or output it in a form directly useful to the user.

The number of such devices or peripherals is vast and the tasks they perform numerous. Many of them also have intelligence of their own and are thus systems within themselves. The question thus arises as to whether they should be included within the definition of a system or not. This decision in general will depend on the type of system analysis for which the particular system definition is being carried out and this again depends on the bounds of the system as decided upon when definition of the system environment is carried out. Once a decision has been made as to which peripheral devices constitute a part of the system the devices must be included in the description of the system.

In the system information is transmitted in terms of multiples of bits or binary words. Thus associated with each device will be an information transfer rate which will be dependent on several factors. Firstly each type of device will have a maximum transfer rate. Secondly transfer of data is also dependent on several actions which could result in this data rate not being transferred continuously but rather in spurts.

When comparing devices this comparison is made in terms of the amount of data transferable in unit time. Thus each individual device will be described in terms of its theoretical maximum transfer rate. This rate is simply the maximum baud rate of the device multiplied by the number of bits transferred simultaneously. For serial devices the value is one whilst for parallel devices it could be any number although typically it is eight or sixteen. For asynchronous parallel devices the baud rate is based on the signal timing of each transfer although no transfer rate is generally given.

The Information Acquisition and Dissemination Particulars are thus expressed in terms of the nominal transfer rates of the devices :

$$Tr_k = B_k \times D_k$$

where Tr_k is the data transfer rate of device k, B_k is the cyclic rate of the device and D_k is the data path width. D is one for serial devices whilst B cannot exceed the baud rate of the device in question although in cases it could be slower.

Determining the maximum value B_n for each device and the actual value B_{np} in unit time gives us :

$$IAP_n = 1 - B_n/B_{np}$$

which expresses the performance of each device.

3.5.4 The System Control and Capability Global

This part of the system description is somewhat conceptual insofar as no physical part of the system is involved. Instead it deals with the manner in which the system's physical elements co-exist and carry out their individual tasks. It also describes those tasks carried out by the system for the user, but which are not a direct part of the user's task.

The first factor of importance concerning this global element is that it in itself consists of a program or set of programs which run on the system, sometimes in conjunction with some hardware contained within the system. These programs are known as the system software. The system software has three main roles. Firstly it must make all decisions concerning allocation of system resources to parts of the workload, such as scheduling of programs to run, allocation of main memory area and allocation of usage of peripheral devices.

Secondly the system software must protect the workload from self destruction or damaging effects within the workload. This aspect, known as system integrity, prevents any one user from prejudicing the execution of another's task. The third system role consists of taking care of the organization and operation of system resources. This includes file management on external memory, peripheral device drivers and numerous other so-called utilities such as keeping tables of information for use by user programs. This aspect can be considered as a service performed for the user.

At this point it is necessary to mention that programs such as compilers, loaders and editors are specifically being excluded. This is contrary to the opinion of some researchers in the field, notably Drummond (1973) who contends that these programs are provided in the system as a service to users much in the same way as other utility programs and are thus not part of the workload but rather part of the operating system. The standpoint taken in this project is, however, based on the premise that these programs are not necessary to system operation and are tasks carried out by the system in direct response to user requests. This approach is supported by Ferrani (1978) who maintains that the workload is any task completed due to some input from outside the system. These utilities are thus part of the workload.

The System Control and Capability Global is one to which it is very difficult to ascribe a performance value and different system software sets could be regarded in a completely different light by different parties. For this reason it is only possible to set down guidelines regarding the important aspects of system software.

1. The more complex a system the more prone it will be to faults.
2. The system software must deprive the workload of as little of the system resources as possible.
3. It must be capable of carrying out all those tasks required of it by the workload.
4. It must be easy to use, this characteristic is generally known as "friendliness", and tolerant to user error.
5. It must protect all tasks and itself from faults occurring in other tasks.

Using these guidelines a nominal performance value for the system software can be decided upon based on various inputs such as a survey of the users, the system manager's experience, records of the usage of the various utilities and the loading of system resources by these utilities. Actual comparative performance can then be determined by the addition or subtraction of elements of this global and depending on the nominal value of the element added or subtracted compared to the whole system control and capability global the performance index will vary, bearing in mind that parasitic effects on the other globals will also play a role.

Expressed mathematically each particular element will have a value expressed by the equation :

$$SCP_n = Uf \times If \times Lf$$

where Uf is the usage of the element, If is the statistically rated importance factor of the element based on a user survey and the system manager or evaluators assessment and Lf is the loading of system resources.

In the case of the SCP no performance measurements are made since an element is either present or not present but the value of the global is increased or decreased by inclusion or exclusion of the elements. It should be noted that in many cases where the system control and capability global is not altered during a performance evaluation study this global will be a constant and can thus be neglected in relative performance studies.

3.5.5 The Parameters Required for System Evaluation

Having developed a theoretical approach to evaluating computer system performance the next step is to evaluate the theory.

From the definitions of the particular parameters it is found that certain of the available quantities are required in order to analyze Computer System Performance, they are :

1. Processor Idle Time.
2. Dynamic Memory Allocation.
3. Direct Memory Access Activity.
4. Input/Output Transfer Activity.
5. Variations in System Control and Capability.

Clearly the acquisition of this data is dependent on the type of computer system involved. The equipment to be used, although it will perform basically the same function for any system, must be compatible with the system or systems in question.

3.5.6 Restraints on Measurement Techniques

It is now known what must be measured and the next step is obviously to develop measurement facilities. However, at this point it is necessary to discuss certain restraints which should be placed on these techniques in order that the data acquired be valid.

Since a computer system is a complex and sensitive machine, interference in its operation by outside sources could conceivably affect performance quite significantly. Measurement techniques must thus be designed to give minimum or preferably zero-loading of the system, and must be invisible to the system in its operation in order that system decisions are not affected. In cases where loading is not zero or invisibility not absolute the effects that do occur must be quantifiable and must be taken into account. These principles must be applied at all times.

4. EVALUATION OF THE COMPUTER SYSTEM ANALYSIS TECHNIQUES

In the previous chapter a mathematical expression was developed which expresses system performance. The actual global elements of performance were expressed in terms of measurable quantities. Gathering the data to be used in the implementation of this theory is the next step.

It is at this stage that the project must swing from a totally general theoretical study to a more specific one. The major reason for this is that, although the measurement techniques which are used are conceptually system independent, in practice, their implementation must allow for the structure of the particular system. Bell and Thorley (1985) conclude, however, after a study of various systems and analysis techniques, that even the most system specific hardware and software tools can be modified to operate on a range of systems and in particular the principles applied are almost always transferable with a modicum of effort.

In this case the system to be studied is the Hewlett-Packard 1000 F minicomputer system of the Department of Electronic Engineering of the University of Natal running under the Hewlett-Packard Real Time Executive IVB software package. The system is used in a research and development role and for computer modeling of engineering systems, whilst it has a secondary role as a real time controller of experiments.

4.1 WHEN IS OPTIMIZATION REQUIRED?

Up to this point system performance optimization and measurement has been discussed in general terms. A question which arises, however, is when is optimization necessary. Obviously a system which is only used occasionally and only carries a small load will not benefit from optimization.

In fact optimization can only really become effective when the system in question starts "hitting the stops"; in other words when one or more elements of the system are being extended to their full range for a significant fraction of the time. Determining whether this happens or not, is not necessarily a simple matter but it is possible to gain a good insight into the loading of the system using a few simple techniques.

Bell an Thorley (1985) suggest three software tools which are always useful to the analyst who wishes to acquire a first order indication of system performance. The first is a CPU soaker which measures CPU utilisation and the others are an I/O logger to log system I/O activity and a CPU sampler to determine CPU tasks at random intervals. The tools used in this study are a CPU usage monitor which corresponds to the soaker and a dynamic usage display which combines elements of the sampler and I/O logger.

4.1.1. The System Usage Monitor and Dynamic Usage Display

These two techniques were developed to attempt to gain an insight into the actual loading of the system. In many systems just such tools as described here are built into the actual system. The techniques described are both software techniques and thus inherently they must affect the loading of the system, but their effect is negligible.

Going back to the definitions of the different global parameters it is obvious that processor throughput is a major factor in the performance of any system. It would thus seem to be a reasonable supposition that processor throughput must give a good indication of system loading, it is after all the system resource most directly coupled to the running of workload programs as well as system programs. The system usage monitor makes use of this fact by measuring the system idle time over a given period, the more idle time measured, the less the loading on the system.

The dynamic usage display plays a different role. This program samples the entire system status at a predetermined interval and displays the status of system memory and all programs active on the system at such times. It also displays the system usage during the previous interval. The program is based on an HP utility program called WHZAT which is a system status display program. An example output from this program is shown in Figure 2.

```
**************************************************
* P£.  SZ.  TP.  PRGM.  TP.   STTS.  PRIOR.  *
**************************************************
*  1    6   RT   SMP     2   DORMNT       30  *
*  2   14   RT                                *
*  3   28   BG                                *
*  4   20   BG   PAS00   3   SCHEDD    32767  *
*  5   14   BG   PAS1A   3   TM LST       44  *
*  6   88   BG M PAS04   3   EXCTNG      100  *
*  7   28   BG C                              *
*  8   20   BG C                              *
*  9   20   BG C                              *
* 10   20   BG C                              *
* 11   88   BG M                              *
* 12   28   BG S                              *
* 13   18   BG S WHZAT  3   DORMNT        41  *
* 14   14   BG S LGOFF  3   GEN WT        90  *
* 15   14   BG S LOGON  3   GEN WT        50  *
* 16   14   BG S PAS87  3   SCHEDD        43  *
* 17   68   BG M                              *
* 18   28   BG S                              *
* 19   20   BG S FMG87  3   GEN WT        90  *
* 20   20   BG S R$PN$  3   GEN WT         5  *
* 21             <Partition Undefined>        *
* 22             <Partition Undefined>        *
* 23             <Partition Undefined>        *
**************************************************
TIME: 15 07 45                  CPU USAGE: 97%
```

Figure 2   Output of dynamic usage display

Once it has been determined that the loading of the system is sufficient to warrant performance optimization the performance analysis techniques can be implemented.

4.2 THE PERFORMANCE EQUATION FOR A REAL SYSTEM

The first step in analyzing a computer system is to formalize the system structure. In this case it was felt that it was beyond the scope of the project to analyze and optimize an operational system. Instead an attempt was made to illustrate the various concepts and techniques and for this reason certain limits were imposed.

Firstly the workload was fixed and consisted of six programs which ran continuously. Secondly only those elements of the computer system which were necessary for operation of the system under the named workload were included in the Computer System Describing equation and lastly only certain dynamically variable parameters within the system were varied to show the technique's capability to register changes in performance, thus enabling the user to optimize system performance accurately for a particular workload.

Appendix D describes the entire experimental methodology followed.

4.2.1 The System Structure

We begin with IPG the processing global which has only one PE since the HP 1000 is a single processor system. Thus :

$$IPG = IPP_1$$

Looking now at the Information Storage Global, ISG, we have two particulars ISP_1 and ISP_2. ISP_1 is the main memory allocation at any time. This is done dynamically by the system and consists of allocating a memory partition to a particular program.

In the HP RTE-IVB operating system memory is allocated in the form of partitions, irrespective of whether a program fills all or only part of the partition. If all programs are thus smaller than the smallest partition available, then the available memory is directly proportional to the number of partitions available. In the tests conducted on this system this was true. Furthermore the total amount of available memory on the system exceeded the maximum requirement by a large margin. The total amount of available memory was thus reduced to fit the exact needs of the tests in order to ensure a controlled situation at all times and was also reduced below the maximum requirement to demonstrate the effects on performance of a shortage of memory. At no time was the size available made larger than the amount required to load all six programs in the workload at one time.

Only three Information Acquisition and Dissemination Particulars were defined, namely three terminals for input/output, running at 9600 baud. Each of these was connected to a standard HP buffered interface card. All three devices were identical and had sequential priorities in the HP 1000's input/output priority structure. It would have been possible to monitor the utilization of each channel individually but this was deemed unnecessary since the figures of interest in this particular case were overall performance of the system rather than individual devices. For this reason IAG was merely regarded as 3 x IAP_1 and measurements were done accordingly, the total number of transfers being recorded rather than the transfers per channel.

The last global which is used in the equation is the System Control and Capability Global. In this exercise it is not, however, necessary to give values to this global since it remains a constant throughout. We can thus define SCG to be 1 at all times since we are not changing the system control and capability.

The entire system is thus :

$$\text{SYSTEM} = \begin{bmatrix} IPG \\ ISG \\ IAG \\ SCG \end{bmatrix} = \begin{bmatrix} IPP_1 & & \\ ISP_1 & ISP_2 & \\ IAP_1 & IAP_2 & IAP_3 \\ 1 & & \end{bmatrix}$$

4.2.2 The Workload Characterization

This is done in two steps. The first step is to obtain the usage factor on the system contributed by each program or type of program.

A software suite was developed for the HP 1000 system which records program activity, reduces the data and calculates the usage factors. It is a system which actually runs on the computer system of interest and so would actually degrade the performance of the system. This is, however, regarded as being negligible. The actual degradation figures were calculated to be less than 2% but even were it significant in terms of performance, the data recorded would still be valid if it is accepted that the system's users in any working day would require a certain set of tasks to be done and if the performance becomes degraded the workload would merely take up more of the system idle time. Thus only when the system is running at full capacity during an entire recording period will the recording program lose any data since any additional load which would have run during the time taken up by the recording software will run outside of the recording time.

Accepting that this data would be valid the entire workload was characterized using this software. The data received from these programs was verified for repeatability using sampling frequencies of 20, 30, 40 and 50 ms and was within one percent for all cases over any measurement period during which the total workload over the period was the same. In the case of the test system the workload consisted of six programs of three types. These programs were drawn up with the objective of excercising the various aspects of the system. They were also intended to be different but the intention was not that a particular program would excersize a particular resource.

The first program type is CRNCH, a program which places a relatively large burden on the processing resources of the system along with loading on the external memory resources. One copy of this program was included in the sample workload. FILE is a program which places a relatively large loading on the input/output resources, together with loading the disc or external memory and the central processor. Two copies of the program were included in the sample workload. RESP is a short program which loads the processor and the input/output resources. Three copies of RESP were included in the workload. All the programs had the same priority.

Using this dummy workload the program significance factors as shown in Table 3 and 4 below were calculated. The reason that there are two sets of data is that when the system structure changes, there can be, as is seen in this case, a shift in the significance of programs due to competition between programs for scarce resources or otherwise. In this case the available memory was artificially reduced with major impact on performance as shall be seen later. Within a single system configuration, however, the repeatability of the measurements was checked and found to be within 1% for all cases, as it should be when the workload remains constant.

Table 3 Program significance factors for system with maximum memory

Program	CRNCH	FILE	RESP
Sig. Factor	0,572	0,266	0,162

Table 4 Program significance factors for system with minimum memory

Program	CRNCH	FILE	RESP
Sig. Factor	0,328	0,354	0,318

Having determined Ug the usage factor for each program, the next step is to look at the need factors Iy_q, which are determined for each program type and particular element in the system.

To implement the system practically, as described in Chapter 3, three types of data are required. The program listing and load map are automatically generated by the system but the program operation histogram is not available. However, a hardware analyzer was required to obtain the data needed for the system performance analysis and this machine was designed to have a histogram output facility. Using this data it was a simple matter to determine the need factors by calculating areas under the relevant parts of the graphs as determined from the load map and listings.

On the following pages are shown the histograms load map and listing of program FILE as an example of the data used to determine the workload constants. Mixed listings are not included due to their length but were

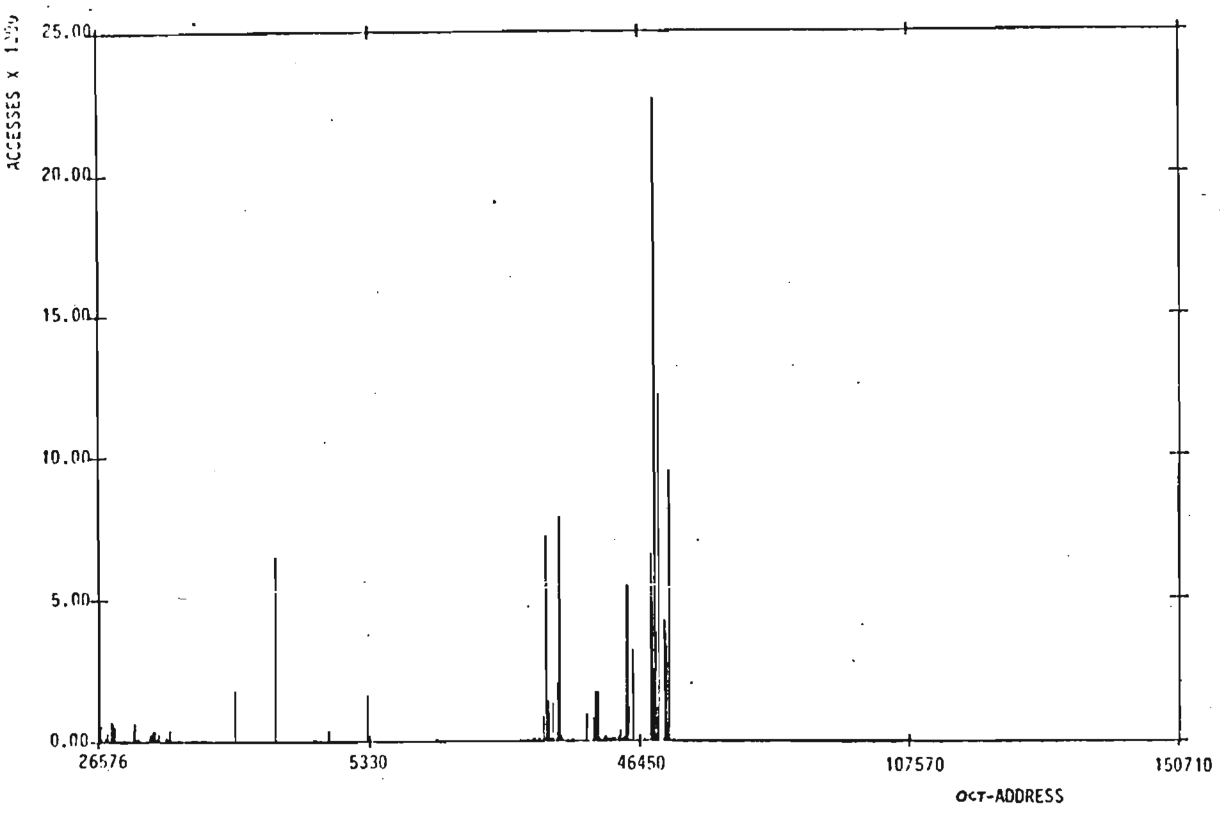


Figure 3 Histogram of operation of program FILE

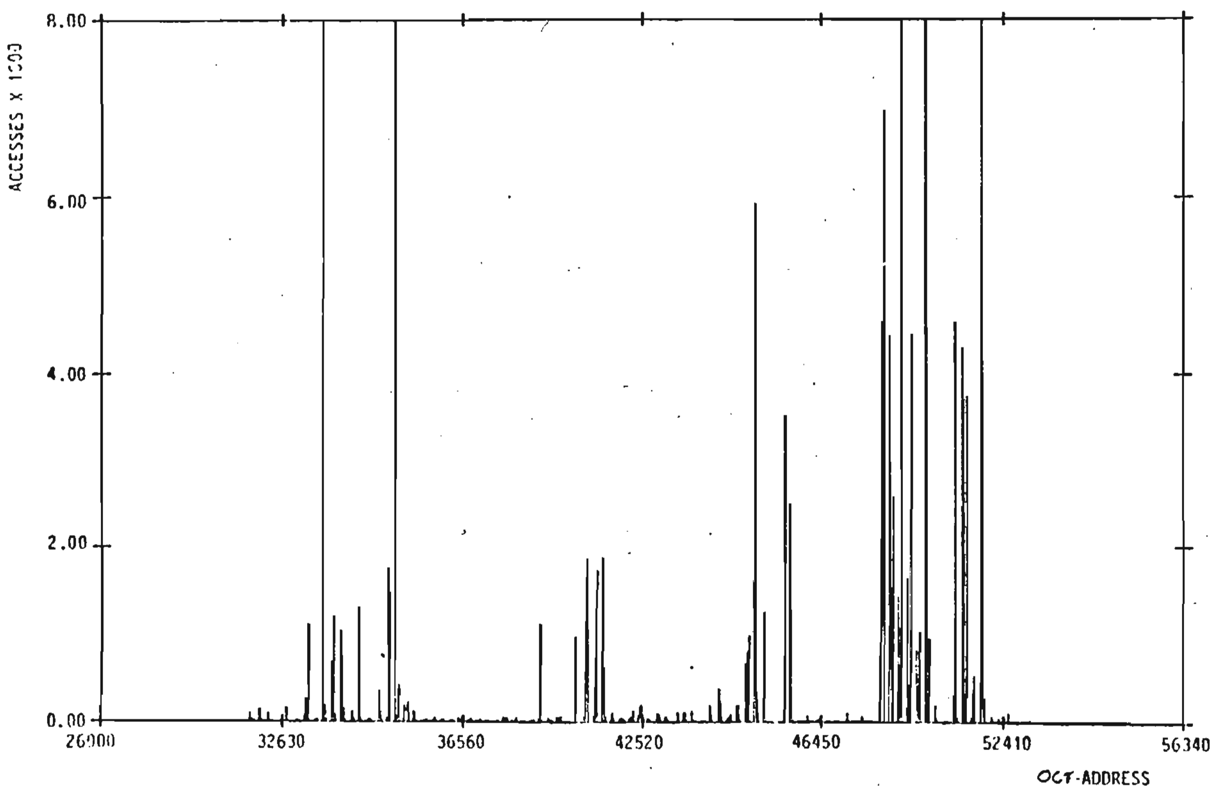


Figure 4 Detail of program FILE histogram

Load map of program FILE

Address

Begin End

```
FILE    32042 37705
FEROR   37706 37750

OPEN    37751 40331  92067-16125 REV.2101 810615
CLOSE   40332 40546  92067-16125 REV.2101 801014
READF   40547 42056  92067-16125 REV.2101 810616
LURQ    42057 42471  92067-1X270 REV.2013 791024
PAUSE   42472 42571  24998-1X253 REV.2101 801007
LUTRU   42572 42700  92067-1X308 REV.2013 790223
OVRD.   42701 42701  92067-16125 REV.1903 780526
.DADS   42702 43011  24998-1X036 REV.2001 780818
.DMP    43012 43157  24998-1X045 REV.2001 780818
RMPAR   43160 43224  92068-1X025 REV.2101 800919
.DDI    43225 43525  24998-1X040 REV.2001 781021
SESSN   43526 43543  92067-16125 REV.1903 780413
R/WS    43544 43702  92067-16125 REV.2101 801013
.DNG    43703 43712  24998-1X046 REV.2001 780818
.DIO.   43713 43774  24998-1X331 REV.2101 800929
.EIO.   43775 45211  24998-1X329 REV.2101 800929
.FMCV   45212 47454  24998-1X333 REV.2101 800709
FMTIO   47455 50706  24998-1X328 REV.2101 800929
.IOER   50707 51022  24998-1X321 REV.2101 800731
.UFMP   51023 51035  24998-1X296 REV.2101 800731
PAU.E   51036 51036  24998-1X254 REV.2001 750701
PNAME   51037 51107  92068-1X035 REV.2101 800919
.DDE    51110 51121  24998-1X039 REV.2001 780818
.DIN    51122 51127  24998-1X042 REV.2001 780818
IFTTY   51130 51215  92067-1X295 REV.2013 790118
$ALRN   51216 51333  92067-1X271 REV.2013 770715
$OPEN   51334 51510  92067-16125 REV.1903 790103
RWSUB   51511 52062  92067-16125 REV.2101 800303
RWNDS   52063 52212  92067-16125 REV.2101 810617
REIO    52213 52337  92067-1X275 REV.2013 790316
$SMVE   52340 52432  92067-1X483 REV.2013 800129
ERO.E   52433 52433  24998-1X249 REV.2001 750701
.OPN?   52434 52457  24998-1X325 REV.2101 800803

10 PAGES RELOCATED    10 PAGES REQ'D    NO PAGES EMA       NO PAGES MSEG
LINKS:BP    PROGRAM:BG    LOAD:TE    COMMON:NC
 /LOADR:FILE    READY AT 10:35 AM  SUN., 10  JAN., 1982

 /LOADR:$END
```

Listing of program FILE

```
FTN4,L,M
             PROGRAM FILE ,3,99
             DIMENSION INAM1(3),IDCB1(144),BUF1(64)
             DIMENSION BUFA(1200)
             DATA INAM1/2H'F,2HIL,2HER/
             LU=1
             TNUM=0.0
1            CALL OPEN(IDCB1,IERR,INAM1,0,0,-52)
             CALL FEROR(IERR)
             DO 9 IC=1,1100
             BUFA(IC)=0.0
9            CONTINUE
             IM=0
25           CALL READF(IDCB1,IERR,BUF1,128,IL)
             CALL FEROR(IERR)
             IF (IL.EQ.-1) GO TO 27
             DO 26 IS=1,IL/2
             BUFA(IM+IS)=BUF1(IS)
             BUF1(IS)=0.0
26           CONTINUE
             IM=IM+IL/2
             GO TO 25
27           IN=0
             CALL CLOSE(IDCB1,IERR)
             CALL FEROR(IERR)
             CALL OPEN(IDCB1,IERR,INAM1,0,0,-52)
             CALL FEROR(IERR)
             TNUM=BUFA(1)
             TNUM = TNUM + 1.0
             BUFA(1)=TNUM
30           DO 31 IX=1,50
             BUF1(IX)=BUFA(IX+IN)
             IF(IN.GT.IM) GO TO 32
31           CONTINUE
             IN=IN+50
             CALL WRITF(IDCB1,IERR,BUF1,100)
             CALL FEROR(IERR)
             GO TO 30
32           CALL CLOSE(IDCB1,IERR)
             CALL FEROR(IERR)
400          FORMAT( " ANOTHER ONE AND " )
             DO 277 IV = 1,200
             WRITE(1,400)
277          CONTINUE
             WRITE(LU,150)TNUM
150          FORMAT( " COMPLETED THE ",F8.1," 'TH RUN "/
            ?" ANOTHER IN 10 SECONDS. ")
             CALL EXEC(12,0,2,0,-10)
             IF(TNUM.GE.10.0) GO TO 99
             GO TO 1
99           STOP
             END
```

used to determine the actual activities carried out by each line of FORTRAN code. In Appendix B a complete set of data is given for all three the programs CRNCH, RESP and FILE.

The only aspect of the of the program resource requirement not obtainable using this technique is the memory requirement. Any program running under the HP RTE-IVB operating system, however, requires at least one partition of memory, independent of program size. Thus a program which only fits into the largest partition is assigned an importance factor of 1. Other programs are assigned importance factors equal to the memory size in pages, divided by the largest partition size in pages. Thus for a program requiring 20 pages with a largest partition size of 80 pages, the importance factor is 0,25.

Using the techniques described and the information given in Appendix B the need factors for the three program types were determined. It should be noted that histograms for various using various sampling rates were analysed and variance over the whole range of sample frequencies was found to be less than 1% in all cases between comparable values for the total number of samples taken.

Table 5 shows the need factors determined for each of the three program types.

Table 5 Need factors per particular element

	CRNCH	FILE	RESP
Ip_1	0,40	0,36	0,57
Is_1	0,14	0,12	0,08
Is_2	0,24	0,16	0
Ii_1	0,02	0,25	0,23
Ii_2	0,02	0,25	0,23
Ii_3	0,02	0,25	0,23
Ic_1	0,11	0,11	0,12

Combining these figures with the significance factors already determined and knowing that the priority factor P is the same for all programs and equal to 1 and further combining the three particular elements IAP_1, IAP_2 and IAP_3 into one, gives a workload description as follows :

$$\begin{array}{ccccc} & \text{CRNCH} & \text{FILE} & \text{RESP} & & \text{TOTAL} \end{array}$$

$$\begin{bmatrix} Kp_1 \\ Ks_1 \\ Ks_2 \\ Ki_1 \\ Kc_1 \end{bmatrix} = \begin{bmatrix} 0{,}228 & 0{,}0958 & 0{,}0923 \\ 0{,}0801 & 0{,}0319 & 0{,}0130 \\ 0{,}1373 & 0{,}0426 & 0{,}0 \\ 0{,}0114 & 0{,}0665 & 0{,}0373 \\ 0{,}1144 & 0{,}0293 & 0{,}0194 \end{bmatrix} = \begin{bmatrix} 0{,}4169 \\ 0{,}1250 \\ 0{,}1798 \\ 0{,}1152 \\ 0{,}1631 \end{bmatrix}$$

for the system configuration with maximum memory.

$$\begin{array}{ccccccc} & & \text{CRNCH} & \text{FILE} & \text{RESP} & & \text{TOTAL} \\ \begin{bmatrix} Kp_1 \\ Ks_1 \\ Ks_2 \\ Ki_1 \\ Kc_1 \end{bmatrix} & = & \multicolumn{3}{c}{\begin{bmatrix} 0{,}1313 & 0{,}1274 & 0{,}1813 \\ 0{,}0459 & 0{,}0425 & 0{,}0254 \\ 0{,}0787 & 0{,}0425 & 0{,}0 \\ 0{,}0066 & 0{,}0885 & 0{,}0731 \\ 0{,}0656 & 0{,}0389 & 0{,}0382 \end{bmatrix}} & = & \begin{bmatrix} 0{,}4399 \\ 0{,}1138 \\ 0{,}1354 \\ 0{,}1682 \\ 0{,}1427 \end{bmatrix} \end{array}$$

for the system configuration with minimum memory.

With these matrices and the global describing functions it is now possible to obtain figures describing the performance of the system relative to every part of the workload and every element of the system, for any variable parameter in the system configuration.

4.2.3 The System Parameters to be Measured

Using the definitions for the structure of the system given in 4.2.1 the various PE are now expressed in terms of system parameters and those parameters which need to be measured are thus identified. Looking first at the Information Processing Particular it is expressed as follows:

$$IPP = 1 - \text{Idle Time/Unit Time} = 1 - It/Ut$$

In the case of ISP the external memory particular the actual allocation of memory was ignored since the available disc memory was far greater than the amount required and this particular was only measured in terms of the transfer rate. Thus :

$$ISP_2 = 1 - Nb \times Bs/Nb_{max} \times Bs_{max}$$

Where the values of Nb and Nb_{max} are the same and Bs_{max} is defined to be 1.14M words per second.

The expression for IAP_1, IAP_2 and IAP_3 is as shown if all three terminals are lumped together and considered as a single device :

$$IAP_1 = 1 - Cx/2880$$

where Cx is the number of transfers made to all three devices since converting the baud rate of 9600 baud for a single device to words per second gives a maximum of 960 words per second for a 10 bit word.

The only other particular not covered at this stage is the main memory allocation. This is done in software in the RTE-IVB operating system in the tables and is not accessible using a hardware monitor. For this reason a small memory resident routine was developed which interrogates the tables and stores this data. This does, however, degrade system performance, but since it was a constant value, the effect was not important in relative measurements of the type conducted in this test. Furthermore, since the allocation is done in terms of partitions or segments, the parameter was expressed in terms of the fraction of segments allocated at any time. For the configuration with maximum memory six segments were available and with minimum memory only one segment.

Thus the particular can be expressed as :

$$ISP_1 = 1 - Nseg/Nseg_{max}$$

It must be noted, however, that in a system where memory size is a critical factor the system employed would not be satisfactory and in fact in virtual memory systems a totally different approach would be required.

The system's performance is now expressed as:

$$\begin{aligned} PSYS = {} & 0,4169(1 - It/Ut) + 0,1250(1 - Nseg/6) \\ & + 0,1798(1 - Bs/1,14M) + 0,1152(1 - Cx/2880) \\ & + 0,1427(1 - 1) \end{aligned}$$

for the system with maximum memory and

$$PSYS = 0,4399(1 - It/Ut) + 0,1138(1 - Nseg/1)$$
$$+ 0,1354(1 - Bs/1,14M) + 0,1682(1 - Cx2880)$$
$$+ 0,1427(1 - 1)$$

for the system with minimum memory.

SCP is in each case equal to 0 since the system control and capability element is constant for all tests.

The structure and workload description are now complete. The next step is to look at the parameters to be measured to determine the system performance. There are four particular parameters which must be measured as listed below :

1. System processor idle time.
2. Main memory allocation.
3. DMA transfer rate.
4. Input/Output transfer rate.

Measurement of these parameters can be achieved by either hardware or software means but the ideal is method is a hardware monitor which would be invisible to the system. This approach is generally accepted and is confirmed by studies carried out by Nichols (1985) who states that the additional complexity in the data reduction to allow for software loading in the target system is virtually impossible to quantify accurately. A hardware analyzer is thus the logical option where possible and in this case such an analyzer the COMPAS or Computer Performance Analysis System with all the required features, as described in Appendix A, was developed. The decision was taken to develop such an instrument rather than purchase a commercially available system, a number of which exist for two reasons. Firstly the commercial systems were all prohibitively expensive and secondly a study of such systems confirmed the opinion of Bekkens and Decauty (1976) who concluded that a dynamically variable analyzer, although complex to use, generally provides a system which far outperforms hardwired or patchboard systems. They also conclude that systems commercially available all fail to provide such facilities. Nichols (1985) agrees with this finding and further lays down a process for determining the required configuration of system. The COMPAS meets and exceeds all the criteria mentioned in this process.

This instrument was used in conjunction with a suite of data reduction programs, which were also developed, to produce the data which was used to analyze the performance analysis theory.

4.3 TEST RESULTS OF A SIMPLE SYSTEM ANALYSIS

Analysis of the system to illustrate the operation of the techniques developed was done in two stages. Firstly the system was configured in the way described previously and results were obtained for various settings of processing time quantum and input/output buffer sizes. The system was then reconfigured with reduced memory and certain tests redone to illustrate the changes produced by reconfiguration of the system resources.

The results obtained from these tests are shown in tabular form in Tables 6, 7, 8 and 9. Figures 5, 6, 7 and 8 show a sample set of graphs for a particular variable setting. full set of graphs are shown in Appendix B. These graphs are extracts of the total set of data recorded for each variable with the no-load situation already incorporated and illustrate the dynamic nature of the data recorded.They do not depict the entire set of readings, numbering aproximately 100 000, recorded for each variable, which were used to calculate the values shown in the tables nor is there a direct time correlation between the graphs. The value depicted in the graphs must thus be subtracted from one to get the actual instantaneous free resource factor in each case. Results 1 to 6 relate to the maximum memory

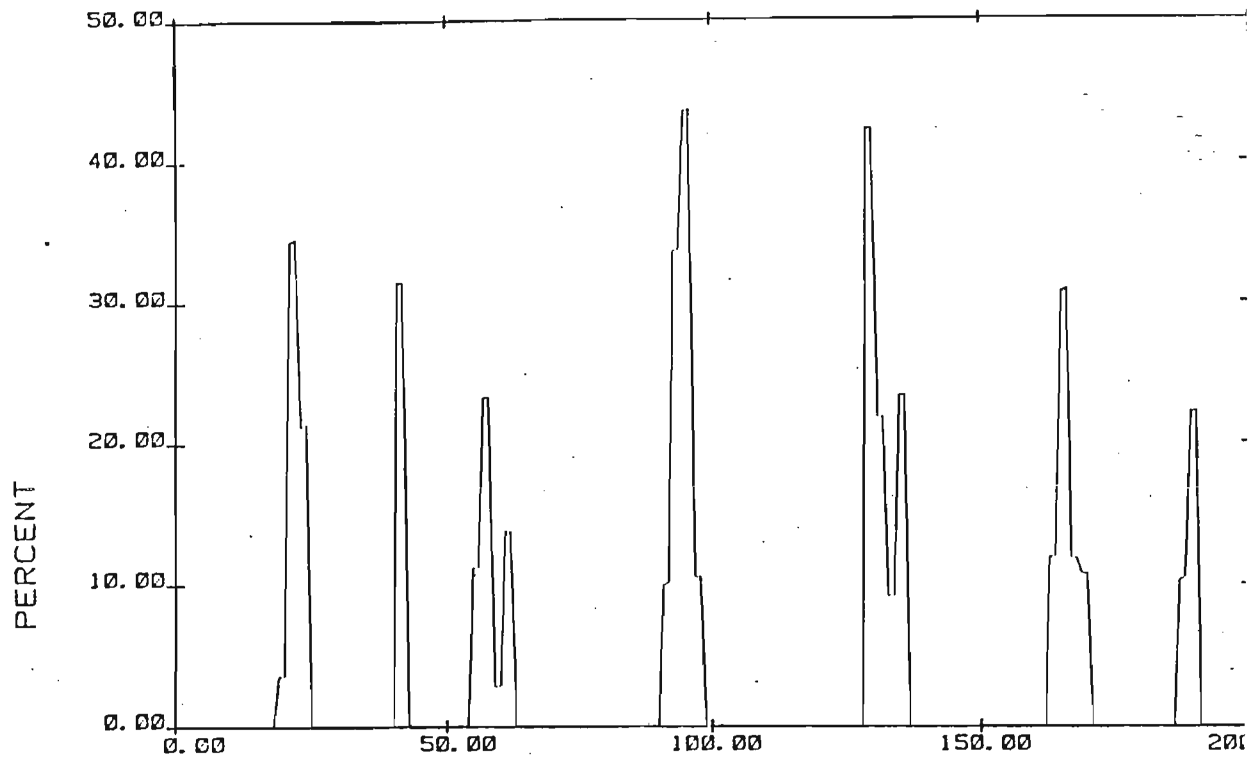


Figure 7 Utilization graph of ISP_2 - Test 1

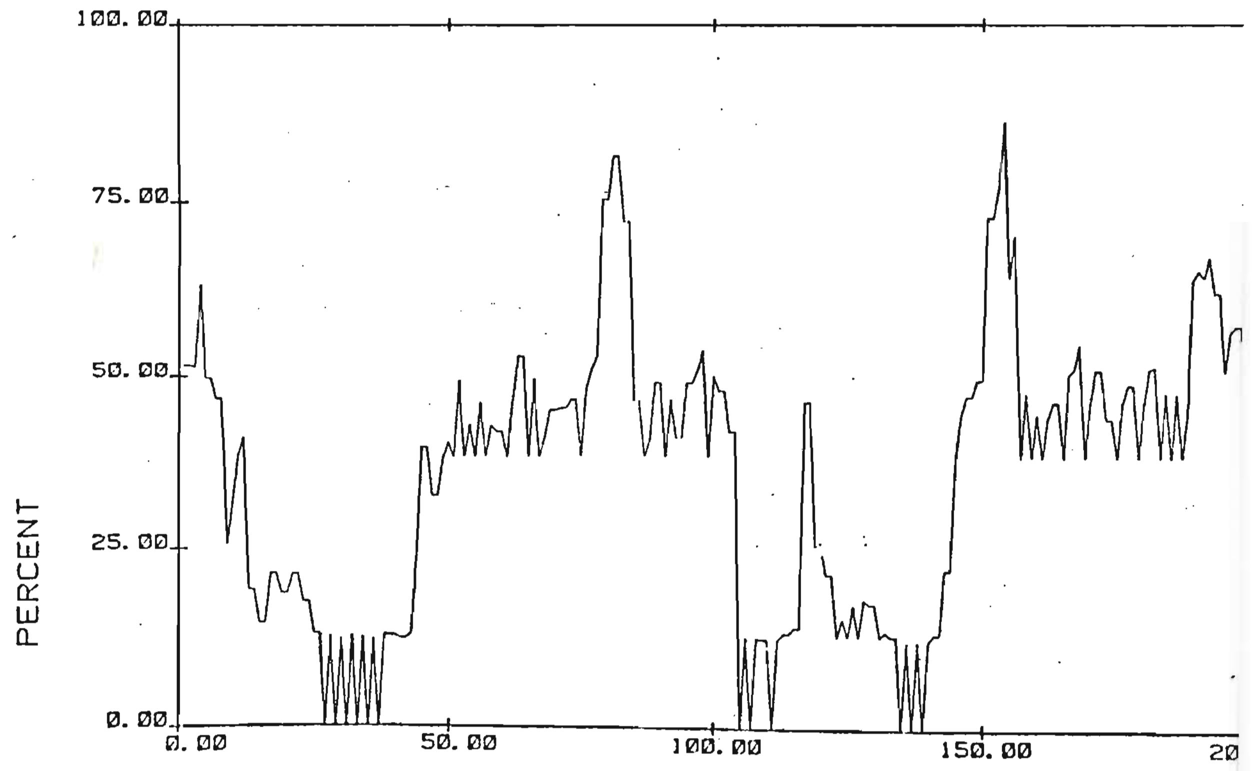


Figure 8 Utilization graph of IAP_1 - Test 1

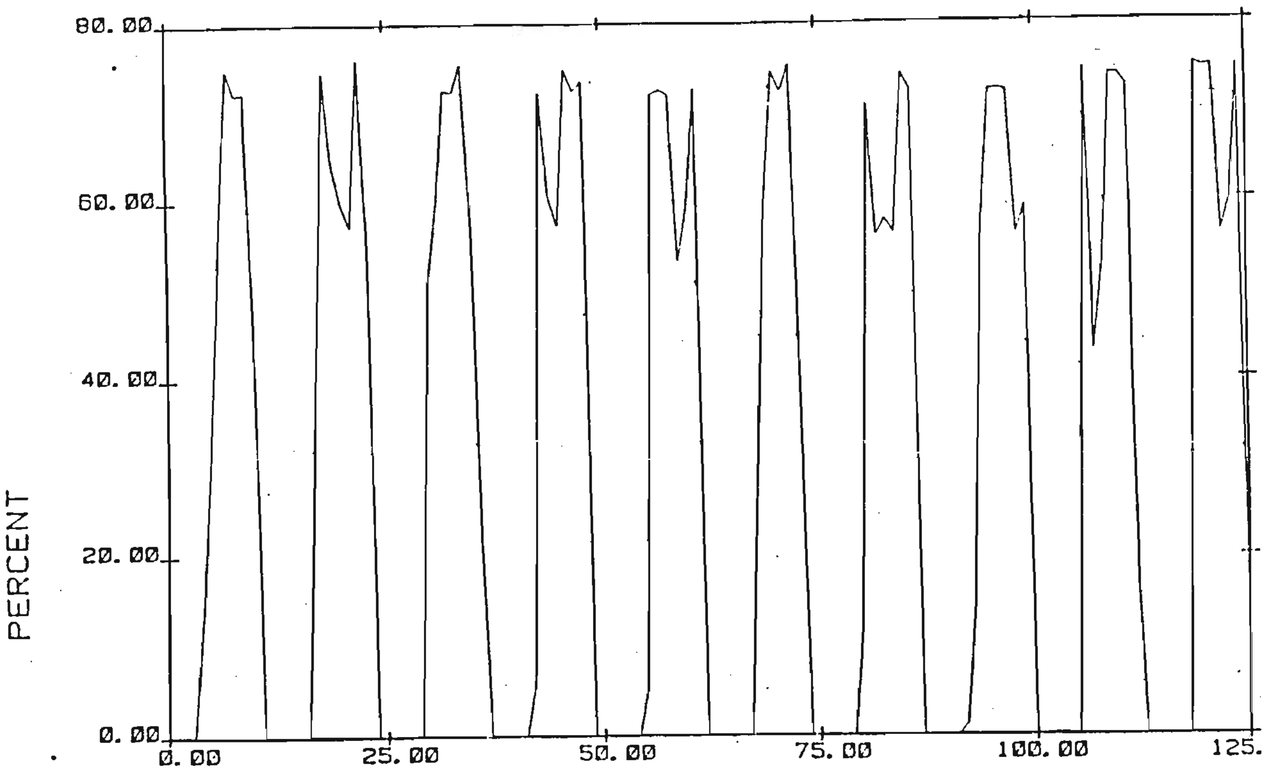


Figure 5 Utilization graph of IPP_1 - Test 1

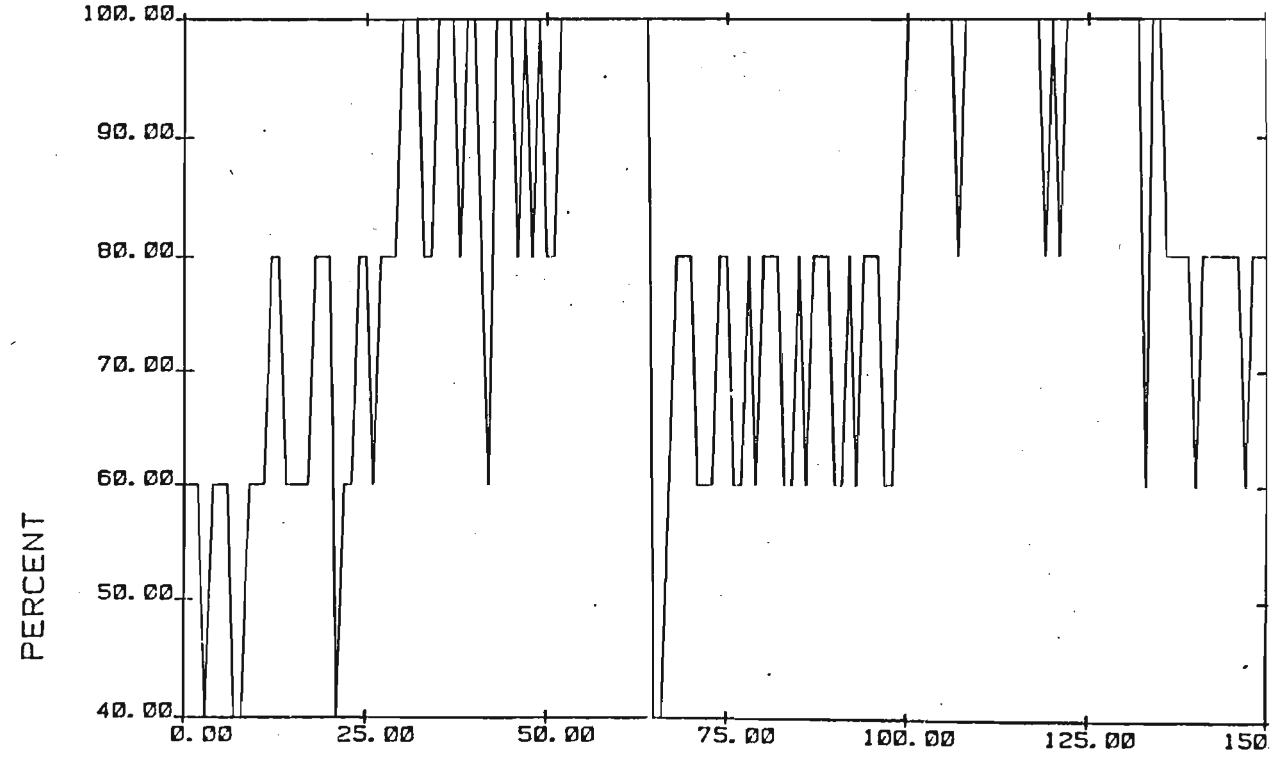


Figure 6 Utilization graph of ISP_1 - Test 1

each case. Results 1 to 6 relate to the maximum memory configuration in each case, whilst 7, 8 and 9 refer to the system reconfigured with reduced memory. The results given were taken with the workload as described over a fixed time period. Each program was run repeatedly at fixed intervals thus giving a constant workload.

A total of nine sets of results were gathered with system configurations as shown in Table 6. The data collected by the system was stored in the form of a large array. The data was then reduced to give a mean value and mean value plus standard deviation for each set of data. These results are shown in Table 7 and Table 8.

Table 6 System configuration and variable settings for test runs

Run No.	Timeslice Quantum	I/O Buffer Limits	Memory
1	50 mS	100,400 words	maximum
2	20 mS	100,400 words	maximum
3	150 mS	100,400 words	maximum
4	50 mS	50,200 words	maximum
5	50 mS	200,700 words	maximum
6	20 mS	200,700 words	maximum
7	50 mS	100,400 words	minimum
8	150 mS	50,200 words	minimum
9	20 mS	200,700 words	minimum

Table 7 Mean percentage usage of system resources

	IPP_1	ISP_1	ISP_2	IAP_1	SCP_1
1	33,08	79,20	4,62	35,60	100,00
2	33,08	78,60	4,52	35,71	100,00
3	33,40	79,50	4,50	35,65	100,00
4	33,19	80,30	4,64	35,61	100,00
5	32,98	80,10	4,52	35,74	100,00
6	32,69	79,71	4,69	35,62	100,00
7	42,36	62,80	35,91	35,74	100,00
8	42,42	64,40	32,37	36,64	100,00
9	46,30	61,30	57,96	35,62	100,00

Table 8 Mean percentage usage of system resources plus standard deviation

	IPP_1	ISP_1	ISP_2	IAP_1	SCP_1
1	50,30	99,36	15,45	56,30	100,00
2	49,82	94,17	13,69	57,38	100,00
3	50,66	99,78	19,56	54,51	100,00
4	51,14	95,26	14,50	55,01	100,00
5	52,34	97,94	18,28	57,23	100,00
6	54,43	98,16	15,94	57,76	100,00
7	52,08	71,62	65,35	48,46	100,00
8	55,68	79,23	59,10	47,15	100,00
9	49,11	65,83	75,24	42,99	100,00

The results depicted in these two tables are the percentage of the available performance for each global type utilized by the workload. The system overheads were not taken into account except where such overheads were directly generated by the workload as is shown by the increased utilizations for the configuration with minimum memory brought about by the need to swap programs in and out of memory. Using these results the mean performance indices were calculated according to the equation developed in section 4.2.4 which states that :

$$PSYS = Kp_1.IPP_1 + Ks_1.ISP_1 + Ks_2.ISP_2 + Ki_1.IAP_1 + Kc_1.SCP_1$$

These results are depicted in Tables 9 and 10.

Table 9 Mean performance index

	CRNCH	FILE	RESP	TOTAL
1	0,5386	0,5795	0,5462	0,5506
2	0,5397	0,5801	0,5464	0,5516
3	0,5372	0,5780	0,5440	0,5491
4	0,5366	0,5777	0,5447	0,5488
5	0,5379	0,5786	0,5457	0,5500
6	0,5393	0,5801	0,5480	0,5515
7	0,4493	0,5153	0,5061	0,4908
8	0,4551	0,5166	0,5024	0,4919
9	0,3828	0,4680	0,4851	0,4455

Table 10 Mean performance index with standard deviation

	CRNCH	FILE	RESP	TOTAL
1	0,4146	0,4242	0,3843	0,4104
2	0,4264	0,4323	0,3953	0,4220
3	0,3998	0,4203	0,3876	0,4030
4	0,4163	0,4309	0,3858	0,4152
5	0,3982	0,4117	0,3717	0,3975
6	0,3950	0,4077	0,3584	0,3921
7	0,3243	0,3832	0,4012	0,3767
8	0,3151	0,3820	0,3908	0,3629
9	0,3222	0,4063	0,4485	0,3922

The figures illustrate the constant performance achieved within a single system configuration. The Performance Index is now recalculated using the mean value plus standard deviation.

The data provided in this section when seen together with the graphical data in Appendix B provides us with a fund of information concerning the operation of the system as well as the needs of the various program types used. Much of this information is clear from nature of the programs, but the high degree of correlation between what is expected and what is measured is significant. A number of the more important aspects are discussed in the next section.

4.4 DISCUSSION OF RESULTS

It must be remembered that the tests done on the system are aimed at illustrating the relevance of the Performance equation and the measurement techniques, and not to attempt to optimize the system for any workload. The sample workload was deliberately kept simple to enable the results to be compared with expected trends based on general knowledge of the computer system and the workload's demands on the system.

For this reason no quantitative conclusions are reached, rather the various aspects of the results are inspected to ascertain whether they conform to predictions based on knowledge of the workload used and the HP 1000 system running under RTE-IVB.

4.4.1 General Loading on System Resources

Looking at the data in Table 7 in conjunction with the graphs in Appendix B three trends are evident. Firstly the mean value of the resource utilization for each of the four significant global parameters for tests sets 1 to 6 are virtually identical. In fact the variation appears to be 1% for ISP_2, IAP_1 and IPP_1 which is within the bounds of the maesurement accuracy of the

COMPAS system. The variation of approximately 2% for ISP_1 is also not unexpected due to the lower measurement accuracy achieved for this global as is shown by the graphs. The resolution of only 20% shown in the graphs is due to the nature of the measurement technique, although the relative accuracy between sets of data was confirmed to be aproximately 2% when the data was averaged.

It is to be expected that these figures would be very close since we are dealing with a situation with fixed resources on which the workload makes fixed demands in a fixed time period. This is due to the fact that the programs in the workload were time scheduled to run at fixed periods during the testing. What is important though is that the graphs of resource utilization against time vary widely showing the dynamic nature of the instantaneous loading on the system, whilst still resulting in the same mean loading over a period of time.

Secondly when we look at the last three sets of test data we find that in two cases the mean value is nearly the same, but in the case of test 9 there is a significant difference. This appears to be due to the fact that one of the resources, the external memory global, was reaching the maximum possible utilization and thus the processor utilization was being degraded due to the processor having to wait for programs to be swapped in and out of memory. The memory utilization also appears to be degraded. This may be explained by the fact that the memory is unassigned whilst swapping is in process. This test was also accompanied by an obvious slowing down in the running of programs during this phase of the testing as displayed by the rate at which the terminal output was updated

The overall performance figures calculated for this test also show that with this system configuration the performance of the system is significantly degraded with respect to the workload in use as shown in Table 9.

Looking specifically at tests 7, 8 and 9 we would expect the very trends displayed when we consider the system parameter settings. If we shorten the time quantum then we get an increased frequency of swapping and this increased frequency of swapping must affect the program which uses the most processing time and external memory, as is shown in Table 10 where the performance index for program CRNCH is reduced by approximately 7% for the test number 9 with respect to tests 7 and 8. For program RESP, which is the least affected this reduction is less than 2%.

The graphs of the various tests also show certain significant patterns. The first is the cyclic nature of system resource utilization which is directly as a result of the programs being time scheduled. The cyclic operation is not, however, evident in the last three tests as it appears to be overridden by the system's need to swap programs in and out of memory. Secondly this cyclic nature is modified in the case of IAP_1 by the buffer size parameter. Particularly in the case of tests 5 and 6 where the buffer limits are changed to 200 and 700 we find that the cyclic frequency is greatly increased and approaches 100% utilization more frequently. This tells us that although the mean utilization is still the same the worst case situation is degraded, something undesirable in the case of an interactive program, for instance, where system response time is important.

Having looked at the resource utilization data in general we now look at the picture which is presented when the workload description is incorporated. We look now at the figures in Table 8. The figures shown are obtained by summing the mean and the standard deviation in each case. The reasoning here is that it is desirable to maintain as constant a loading on the system as possible. Thus when seen in conjunction with the mean value the best case would be the one where the mean value is as highe as possible whilst the standard deviation should be as low as possible, indicating a more constant loading on the system. The standard deviation is a measure of the variation in the recorded data and this sum is a good general indication of the instantaneous worst case utilization of any resource. The use of mean value plus standard deviation as a measure of instantaneous variation in loading is a well recognized statistical technique and is strongly supported by Jack (1985) who contends that this measure is in fact often more relevant in performance studies than the use of simple mean values. He states that this measure is particularly important since a high standard deviation immediately conveys to the analyst the possibility of serious problems. When the values are further combined with time related graphical data, as is the case here, it is also generally possible to identify these problem areas, which leads to a far clearer understanding of the critical areas in system performance.

Mean value plus standard deviation is also used in this case to find the best parameter settings between two or more cases where the means are substantially the same since a more constant utilization of resources is deemed desirable.

It should be noted that in the case where the mean value for a parameter begins to approach the maximum level the standard deviation for the parameter will be substantially affect by skewing and thus become somewhat less relevant. For the purposes of this study these figures which incorporate the standard deviation are thus most valuable when looking at tests 1 to 6.

4.4.2 System Performance with the Test Workload

We now move more specifically to a discussion of the system performance as it appertains to the sample workload. In Table 11 the results for Performance Index with Standard Deviation are rated in order from best to worst for each program and the entire workload. These figures when seen in conjunction with Table 10 allow us to analyze the performance of the various parameter settings and workloads. The table shows the order of merit by test number for the tests done with maximum memory. The values of the timeslice quantum parameters in mS and the buffer limit settings showing lower and upper bounds are also given.

Using this table it is possible to draw certain conclusions. Firstly for a buffer parameter setting of 50/200 or 100/400 words the performance index is

Table 11 Performance index in order of merit

	CRNCH	FILE	RESP	TOTAL	TIME mS	BUFFER low/up)
1	2	2	2	2	20	100/400
2	4	4	3 (150; 100/400)	4	50	50/200
3	1	1	4 (50; 50/200)	1	50	100/400
4	3	3	3 (50; 100/400)	3	150	100/400
5	5	5	5	5	50	200/700
6	6	6	6	6	20	200/700

better for a shorter time quantum. Although in the case of RESP this does not appear to be true we see the results are spread over a span of only 0,3%, well within the resolution of the system, namely 1% overall, thus the fact that test number three is second best could conceivably be an error. Furthermore in all cases, looking at Table 10 we see that, although test 2 appears to give the best result, tests 1, 3 and 4 are all very close whilst 5 and 6 are clearly poorer.

From the results it can be deduced that the system was very sensitive to the larger buffer size especially in the case of RESP with a 20 mS timeslice quantum. Secondly in all cases the very short 20 mS timeslice gave the best result with a smaller buffer size, whilst conversely, the worst result with a larger buffer size.

A possible reason for this is that the larger buffer size resulted in a higher cyclic frequency in the information acquisition and dissemination global IAP_1, thus causing a less smooth utilization of the other resources and consequently higher peak utilization. The causes for this are tied in with the way in which the system makes use of these buffer limits, but are in fact irrelevant to this study. What is important is the technique's ability to register these factors.

Looking now at Table 9 we can also see that CRNCH is the most sensitive when it comes to the reduced memory situation, particularly since it is a program which places a higher demand on the external memory global as well as the fact that it would be swapped the most since the total processing time for CRNCH is the longest. Conversely RESP is less affected as would be expected.

From the results it is thus possible to see that for the test workload test number 2 appears to produce better results which is in line with the conclusion one would make based purely on a knowledge of the systems operation. It can also be deduced that a strong possibility for further improvement exists if the system were to be reconfigured with a timeslice quantum of 20 mS and buffer limits of 50/200 words.

5. DISCUSSION AND CONCLUSION

A need has been identified for a practical and usable method to study computer system performance. In the first three chapters a mathematical model was developed to describe a computer system and its performance, whilst in the latter part of this document a description is given of the measurement techniques which were developed and the results achieved using these techniques in an attempt to illustrate the relevance of the model.

The technique was aimed throughout at a practical approach and is based on the simplest possible premises and arguments. The result was a very straightforward, and at times simplistic, but a very relevant product. Applying the technique is well within the capabilities of any person with reasonable knowledge of computer systems and this is seen as its main strength when compared to the other methods used for computer system performance evaluation.

Furthermore, as was stated previously, the technique can display any aspect of the system and its performance in context with the operation of the rest of the system. The significance of this was very well displayed in the final test where the external memory utilization reached saturation and resulted in degradation of the processor utilization, a factor which would not necessarily be displayed by other methods of analysis which do not follow an integrated approach.

The measured workload techniques also proved to be successful, in that using measured data the mathematical structure of the workload varied dynamically according to the system configuration, as was illustrated in chapter 4.

There are however certain weaknesses evident in the system which could bear further investigation. Firstly the question of determining memory usage on the system. As was stated previously, a software technique was used which did impact the system, although in this case the effect appeared to be insignificant. The system was not compatible with virtual memory systems, however, and also did not fit in with the aim for a totally invisible measurement system.

A possible avenue of investigation which can be suggested and has possibilities is centered around the fact that the majority of multi-user operating systems are table driven. These tables reside in memory at specific locations. If they could be written to two banks of memory, simultaneously, one of which could be read by the measurement system directly, then this problem could be erased. Secondly in virtual memory systems it might be possible to relate paging exceptions in such a system to memory requirements, a technique which is widely used in the large mainframe environment nowadays.

Another aspect of the system which also relates to the measurement techniques, which would be enhanced by direct access to the tables, is the workload recording techniques. Here again drawing program run data directly from the tables would result in the measurement technique becoming totally invisible to the system with all the resultant advantages in reliability and the removal of the resource loading on the target system.

Lastly, although the present system gathers data adequately it would be possible to develop the COMPAS system and the surrounding software suite to reduce the complex process which at present accompanies use of the system significantly.

It was found that attaching the COMPAS to the system did result in a significant increase in system faults, directly due to the need to load the bus of the HP 1000F which was not designed for this. This is a general problem with hardware analyzers as found by Nichols (1985) and can only be avoided if suitable tapping points were provided by the manufacturer. This is, however, unlikely as there is not yet sufficient motivation for manufacturers to move in this direction.

Secondly with the system in the form it was designed, the operator was required to go through a lengthy set-up process for each set of results. This aspect could be improved upon by enhancing the COMPAS software as well as the link with a secondary computer. In this project it was not, however, regarded as necessary due to the highly experimental nature of the technique. The result was that recording the nine sets of test data took in excess of seventy two hours of which only a small fraction was expended on the actual test runs, whilst the majority was used in setting up the COMPAS system and reducing the data to a usable form, the 45 final values found in Tables 7 and 8 being produced from approximately 4 500 000 raw data points while the 40 workload matrix values were produced from approximately 16 000 000 raw data points.

Another area which is seen as one which requires further investigation is the determination of program type definitions and resource requirement specifications. Although the proposed technique appears to work adequately, it is grounded on very loose principles, and was developed to illustrate the computer system performance analysis technique. Further investigation could produce more rigid methods which would reduce the subjective content of the final results. The proposed technique is nevertheless regarded as being relevant insofar as it was used in a study of relative performance and any inaccuracy in this assessment of resource requirements would tend to be canceled out.

Looking now to the positive aspects of the project, certain aspects stand out. Firstly, as was mentioned, the matrix-like structure certainly clearly illustrates the system configuration as far as performance is concerned. Secondly the integration of the workload into this structure makes use of the results obtained simple, and clearly illustrates the effect of changes in the system configuration. The measured workload technique is also very important due both to the accuracy of the results and the fact that it is based on very simple techniques. The ability to display dynamic variations on the workload with changes in system configuration is also a very important advantage.

The accuracy of the measurement techniques as well as the fact that they provide a clean picture of the dynamic form of the resource utilization is also of great value.

In conclusion it is felt that the study undertaken was both worthwhile and successful and holds great promise. The goal of developing a relevant structure for performance analysis as well as the techniques for gathering the necessary data were successfully completed and great possibilities for future developments in this field have been opened up.

6. REFERENCES AND BIBLIOGRAPHY

BARD, Y., 1976 A characterization of VM/370 workloads. Proc. Workshop on Modeling and Performance Evaluation of Computer Systems. The Commission of the European Communities, Joint Research Centre, Vorese, Italy.

BEKKENS, Y. and DECAUTY, B., 1976. A versatile programmable hardware monitor. Proc. Workshop on Modeling and Performance Evaluation of Computer Systems. The Commission of the European Communities, Joint Research Centre, Vorese, Italy.

BELL, T.E. and THORLEY, T.M., 1985. Minicomputer performance analysis problems and techniques. CMG '85. Proc. International Conference on the Management and Performance Evaluation of Computer Systems. Computers and measurements group, Alexandria, Virginia.

BELL, T.E. and THORLEY, T.M., 1985. Tools to analyze minicomputer performance. CMG '85. Proc. International Conference on the Management and Performance Evaluation of Computer Systems. Computers and measurements group, Alexandria, Virginia.

BOROVITS, I. and NEUMANN, S., 1979. Computer Systems Performance Evaluation, Criteria, Measurement, Techniques and Costs. Lexington Books. Toronto.

CROFT, F.M. and CANTRELL, R., 1986. Performance measurement applied to large scale interactive computer systems. Computers and Industrial Engineering. Vol 12, 1987. Pergamon Journals, Great Britain.

DRUMMOND, M.E., 1973. Evaluation and Measurement Techniques for Digital Computer Systems. Prentice-Hall Inc., Englewood-Cliffs, New Jersey.

FERRANI, D., 1978. Computer Systems Performance Evaluation. Prentice-Hall Inc., Englewood-Cliffs, New Jersey.

HAYES, J.P., 1978. Computer Architecture and Organization. Mc Graw-Hill Kogakusha, Ltd., Tokyo.

Further studies along the lines proposed in this project could result in very significant improvements in the performance of computer systems as well as leading to a clearer understanding of the operation of complex computer systems.

HEWLETT-PACKARD, 1977. HP 1000 F Operators Reference Manual.

HEWLETT-PACKARD, 1979. HP RTE-IVB. Programmers Reference Manual.

HEWLETT-PACKARD, 1979. HP RTE-IVB. Terminal Users Reference Manual.

JACK, H.E. 1985. Some parameters to evaluate performance and and capacity on Burroughs B5000/B6000/B7000 mainframe computers. CMG '85. Proc. International Conference on the Management and Performance Evaluation of Computer Systems. Computers and measurements group, Alexandria, Virginia.

KOBAYASHI, H., 1978. Modeling and Analysis : An Introduction to System Performance Evaluation Methodology. Addison-Wesley Publishing Company. Reading, Mass.

KUCK, D.J., 1978. The Structure of Computers and Computation Vol. I. John Wiley and Sons, New York.

LAZOWSKA, E.D., ZAHORJAN, J., GRAHAM, S., SEVCIK, K., 1985. Quantitative System Performance: Computer system analysis using queueing network models. Prentice-Hall Inc., Englewood-Cliffs, New Jersey.

LORIN, H., 1982. Introduction to Computer Architecture and Organization. John Wiley and Sons, New York.

MAC NAIR, E.A. and SAUER, C.H., 1985. Elements of practical performance modelling. Prentice-Hall Inc., Englewood-Cliffs, New Jersey.

MYERS, G.J., 1982. Advances in Computer Architecture, Second Edition. John Wiley and Sons, New York.

NICHOLS, S.R., 1985. A decision process for hardware monitors. CMG '85. Proc. International Conference on the Management and Performance Evaluation of Computer Systems. Computers and measurements group, Alexandria, Virginia.

SVOBODA, L., 1976. Computer Performance Measurement and Evaluation Methods : Analysis and Applications. Elsevier Scientific Publishing Company, Amsterdam.

APPENDIX A : The COMPAS Computer Performance Analysis System

In order to make measurements of dynamic performance parameters on the system a method must be found to extract the necessary data from the system without affecting the system operation. The areas of interest are processor throughput, peripheral device operation, memory utilization and DMA activity.

Measuring processor throughput requires knowledge of two aspects of its operation. Firstly the number of instructions executed in unit time and secondly the function carried out by specific set of instructions. Peripheral throughput requires knowledge of the number of transactions in unit time and the actual time during which such transactions are possible, the interruptability of a particular peripheral device.

The number of instructions executed should ideally be measured on the processor. This necessitates modification to the hardware in practice, since the signals of interest are totally confined to the processor board in question. An alternative is to record the number of instructions fetched from the memory. This approach is far easier to accomplish as the memory buses are easily accessible on the HP 1000 system. Task recognition is also fairly simple assuming that the address bus of the memory is available.

The technique used is to count the number of accesses to memory to compute the number of instructions executed. This is not entirely accurate since many instructions include a secondary or even multiple memory references in their execution. Assuming, however, that a standard instruction mix is valid for the processor, it is a simple step to assume that instructions executed will normally occur in a fixed proportion to the recorded number of memory accesses. If this is true then relative measurements will be valid.

Insofar as task recognition is concerned, this is achieved by recording the actual memory address accessed. If a memory map of the system is now available, and this is so, it is simple to relate a sequence of memory accesses to the execution of a particular task.

In the use of the peripheral elements on the HP 1000 system it is true to say that each individual transaction, except in the case of DMA transactions, generates a single interrupt. Now on the processor it is possible to access the interrupt priority and from this chain it is possible to record directly the occurrence of a device interrupt as well as its duration and in addition the device's interruptability can also be recorded directly. To monitor the DMA transfers a further signal is available on the backplanes, which indicates the occurrence of a DMA cycle.

It is thus possible in theory to obtain all the data required to calculate the various global performance parameters. The problem is not quite so simple though as the cycle times being dealt with are so short, in the region of 2 seconds, that some form of recording and subsequent evaluation is required. An oscilloscope or logic analyzer could be used, but the amount of data which can be observed in this manner is very limited. A device is thus required which can observe and record the data at rates comparable to the speed of operation of the actual system.

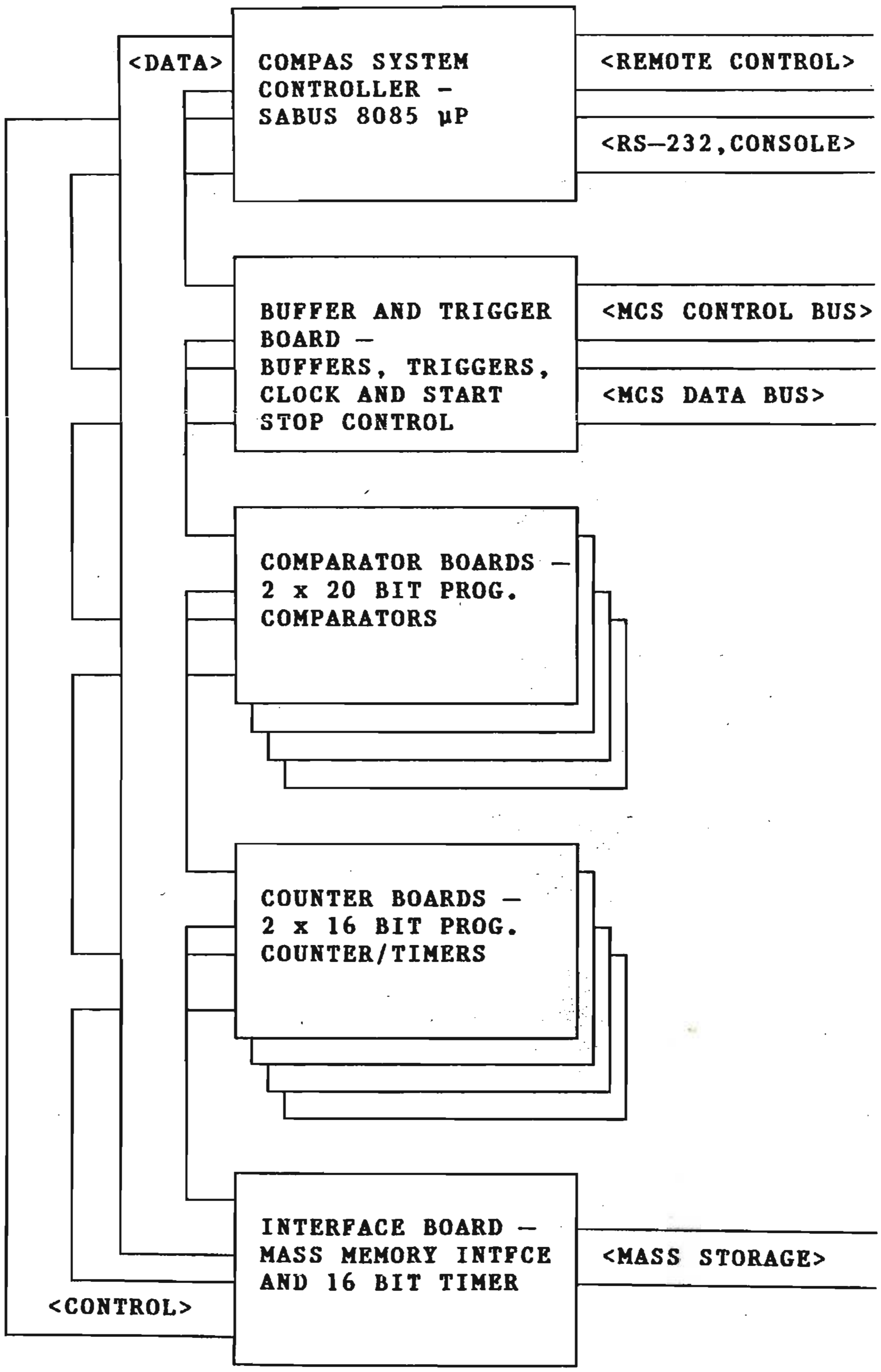
A number of commercial systems are available but suffer from certain deficiencies. All appear to be aimed at hardware monitoring and only two include optional comparators. Furthermore they require manual reconfiguration. Lastly price is an important factor and in this country availability is also a problem. The only similar machines readily available in this country being very akin to logic analyzers and very expensive.

For this reason an experimental system to perform the required tasks was designed, constructed and tested. The Computer Performance Analyses System or COMPAS as it is called, is divided into five operational blocks. It consists firstly of the microprocessor based control module, a buffer board which handles all the signal conditioning from the system being analyzed, comparator boards which are used to select data for recording, counter/timer boards which record number and duration of events and external interface boards which handle the storage of data on external mass memory.

The design of the prototype, a block diagram of which occurs on the next page is able to include up to three comparator boards and three counter boards. The system is designed to be system independent and only the buffer board need be modified when various systems are analyzed. The basic concept is also expandable to include internal memory up to 64 K words and the system can be used as a stand alone monitoring system run interactively from the system console or it can be configured to run under the control of some external computer system, even the system being monitored.

The Control Module

The COMPAS has to deal with cycles times of 1 to 2 µseconds and so no microprocessor commonly available can do the entire task under software control. For this reason a compromise has been made. The processor, under software control, configures each module in the system according to the actual task it is required to perform when recording the data and then allows each board to run independently as configured once recording begins. The timing goals can thus be met.



<DATA>
COMPAS SYSTEM CONTROLLER - SABUS 8085 µP
<REMOTE CONTROL>
<RS—232,CONSOLE>
BUFFER AND TRIGGER BOARD — BUFFERS, TRIGGERS, CLOCK AND START STOP CONTROL
<MCS CONTROL BUS>
<MCS DATA BUS>
COMPARATOR BOARDS — 2 x 20 BIT PROG. COMPARATORS
COUNTER BOARDS — 2 x 16 BIT PROG. COUNTER/TIMERS
INTERFACE BOARD — MASS MEMORY INTFCE AND 16 BIT TIMER
<MASS STORAGE>
<CONTROL>

Block Diagram of COMPAS System

Once recording ends the processor then reads the data off the modules concerned and stores it in its own memory or retransmits it to some external master system, as and when required. The actual controller in this case is a standard SABUS 8085 single board computer commercially available. It includes an RS232 serial port configured as the system console, an 8255 parallel port used as a secondary port for external control of the system, 1 K of RAM and up to 16 k of ROM for the COMPAS software to be stored in.Within the system each function is addressed by the control processor using an 8 bit function address. This address consists of a five bit switch selectable address for each board plus a fully decoded 3 bit address for each board function. This allows a maximum of 32 boards with 8 functions on each.

The Buffer/Trigger Board

This board buffers the incoming signals to reduce the signal loading experienced by the monitored computer system (MCS) as far as possible. This board has several additional roles as well. It conditions the incoming trigger signals from the MCS which in the case of the HP 1000 is a double pulse to produce a single trigger pulse for the COMPAS. A secondary trigger is also produced by dividing down the primary trigger and this is used in cases where data sampling is required. The secondary trigger frequency can be programmed to vary from equal down to one sixteenth of the primary trigger frequency. The buffer trigger board also produces a programmable clock frequency varying from 18 MHZ to 475 KHZ in factors of 2.

In addition the eight qualifier input signals can be inverted and combined programmably in any combination to produce three COMPAS system qualifiers.

Lastly the system on/off and halt control flip-flops are built into the buffer/trigger boards. These flip-flops allow the entire system to be stopped and started by the control module as well as including a halt facility whereby any individual module can halt the system if certain conditions such as counter overflow occur.

The Comparator Boards

The prototype is designed to accommodate three comparator boards although in fact only one is actually included. Each comparator board contains two 20 bit comparators for address comparisons on the data input from the MCS memory address bus. The outputs of the comparators, designated A and B, are then decoded to give outputs of data equal to A, equal to B and equal to or between A and B. These outputs are then conditioned to allow qualification by the secondary trigger or any of the system qualifiers in any combination.

The final outputs can then be addressed individually to any one of eight or tieable lines of the system data true bus, this bus being used to transfer signals between modules of the COMPAS. These outputs can also be configured to trigger the system halt circuitry.

The comparison reference addresses are stored on the comparator boards in latches by the control board and can be configured to any values required.

The Counter/Timer Boards

Each of the up to three counter boards configurable in the prototype system contains two 16 bit counters. These counters are individually triggered by one of three trigger signals programmed by the control module from any combination of the system trigger, the secondary trigger and the three system qualifiers as well as any one of the signals on the system data true bus. In addition they can each be programmed to act as a timer using the programmable clock signal generated on the buffer board, with one of the three trigger signals starting the timing and a second stopping it.

The outputs of the timers can then be read by the controller once recording ends or they can halt the system when either one overflows. In the prototype only a single counter board in included.

The Interface/Timer Board

The primary role of this module is to interface to external memory to facilitate the storage of the recorded memory accesses. In the case of the COMPAS prototype this external memory is in fact an HP 1000 M processor. This processor contains 32 K words of memory and of this 16 K words is used to store data, the data being transferred from the interface to the M processor via a 16 bit parallel interface board under interrupt control, or, for high speed transfers using the DMA facility of the M processor.

As the COMPAS uses a 20 bit internal memory address data bus and the external memory is only 16 bits wide the interface board also has to convert the 20 bit data to 16 bit data. This is done by transmitting under program control any adjacent 16 bits of the actual data word, that is either bits 0 to 15 or 1 to 16 or 2 to 17 and so one. The actual transmission of data is triggered by any one of data true lines or the secondary trigger and any number of these lines can be selected. In addition two of the data true lines can be selected to act as start and stop control lines whilst one of these can start and stop the transmission.

In addition to these functions the interface board also has on it a 16 bit programmable timer which is triggered by the system clock and can be used optionally to halt the system after a preprogrammed interval has elapsed.

The COMPAS Mainframe

All the boards comprising the system are designed to plug into a racking system. This racking system has a split backplane with both SABUS backplane connectors for use with SABUS compatible modules and a dedicated bus designed specifically for the COMPAS. In addition the rack has a front panel which displays the most important COMPAS signals as well as decoding and displaying, on a variable range 64 LED display, the area of memory being accessed at any moment, with resolution down to any two page area. Certain interrupt driven functions of the COMPAS can also optionally be implemented from front panel push buttons, giving limited front panel operation. The system AC and DC power switches and the system reset are also on the front panel. The software is written in such a way that the front panel control features can be disabled.

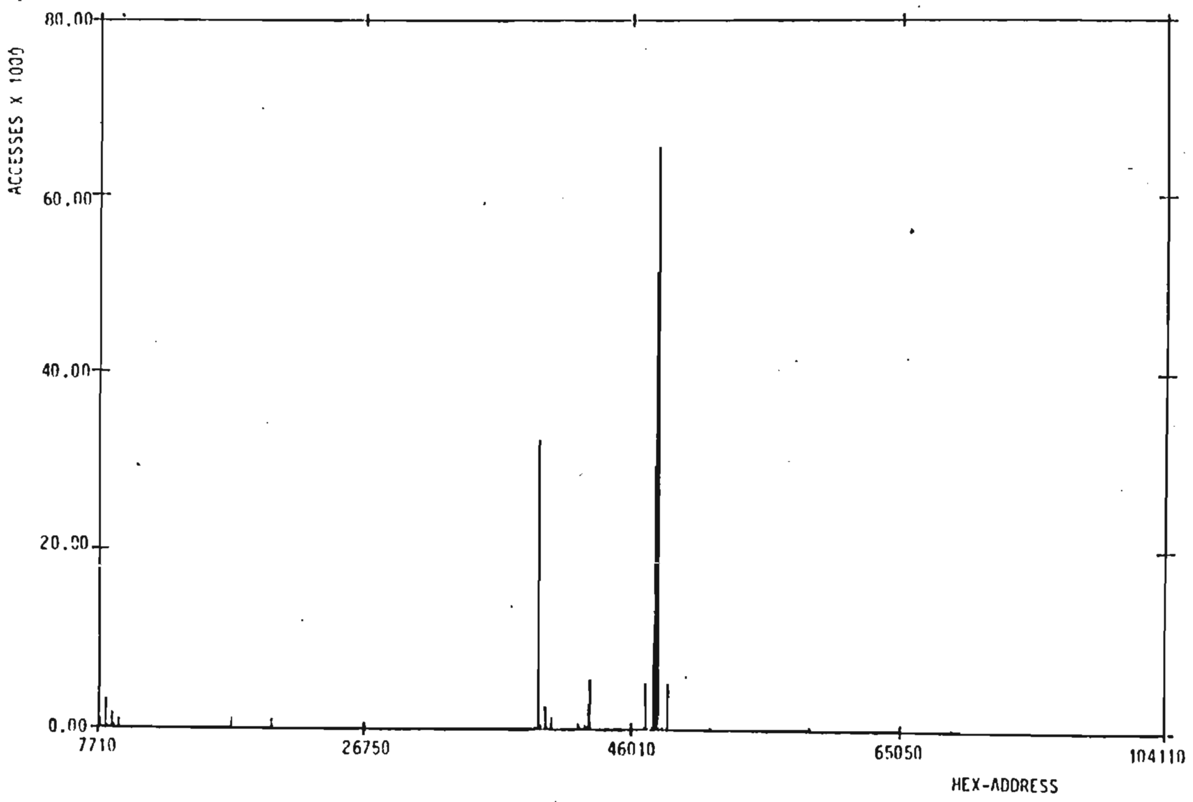
The back panel of the rack contains all the connectors for the various external devices. These are the RS232 port for the system console, the data input connectors, the controller parallel port and the interface board output, as well as connectors for external DC supplies. The external DC supply is in fact used in the prototype at present as an internal supply has not been installed due to cost. A backplane extender connection for the COMPAS bus is also available allowing expansion of the system to a second rack, or the use of logic analyzers or oscilloscopes to display internal signals. At present the AC supply is only used to drive two cooling fans mounted in the back panel.

The COMPAS Operating System

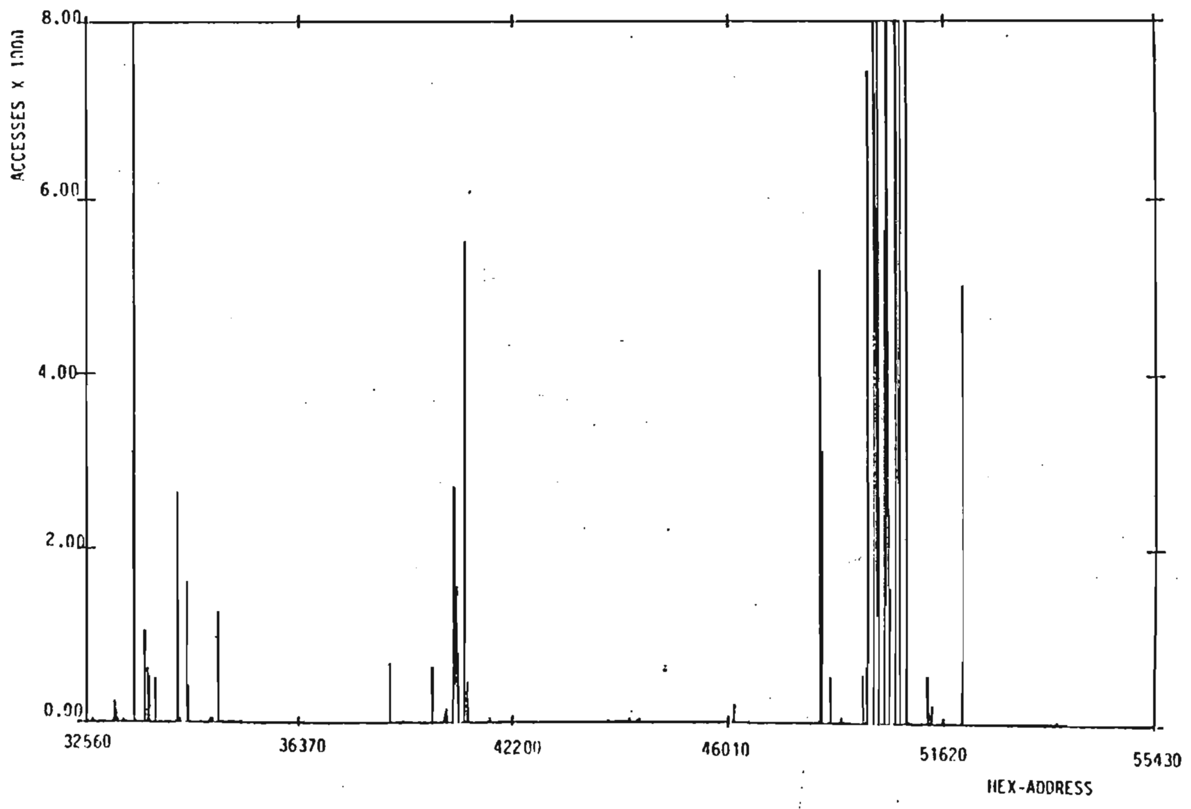
This is a 16 Kilobyte program which was developed to enable the system to be used. It has facilities to configure and run the COMPAS system as well as retrieving, storing and re-transmitting the data. It is a menu driven system which can be operated from the system console or it can be programmed to run from a remote computer.

Once the COMPAS system had been developed and tested it was used in conjunction with the program suite developed to test the Performance Evaluation theory on the Department's HP 1000 system.

APPENDIX B : WORKLOAD CHARACTERIZATION DATA



Histogram of operation of program CRNCH



Detail of program CRNCH histogram

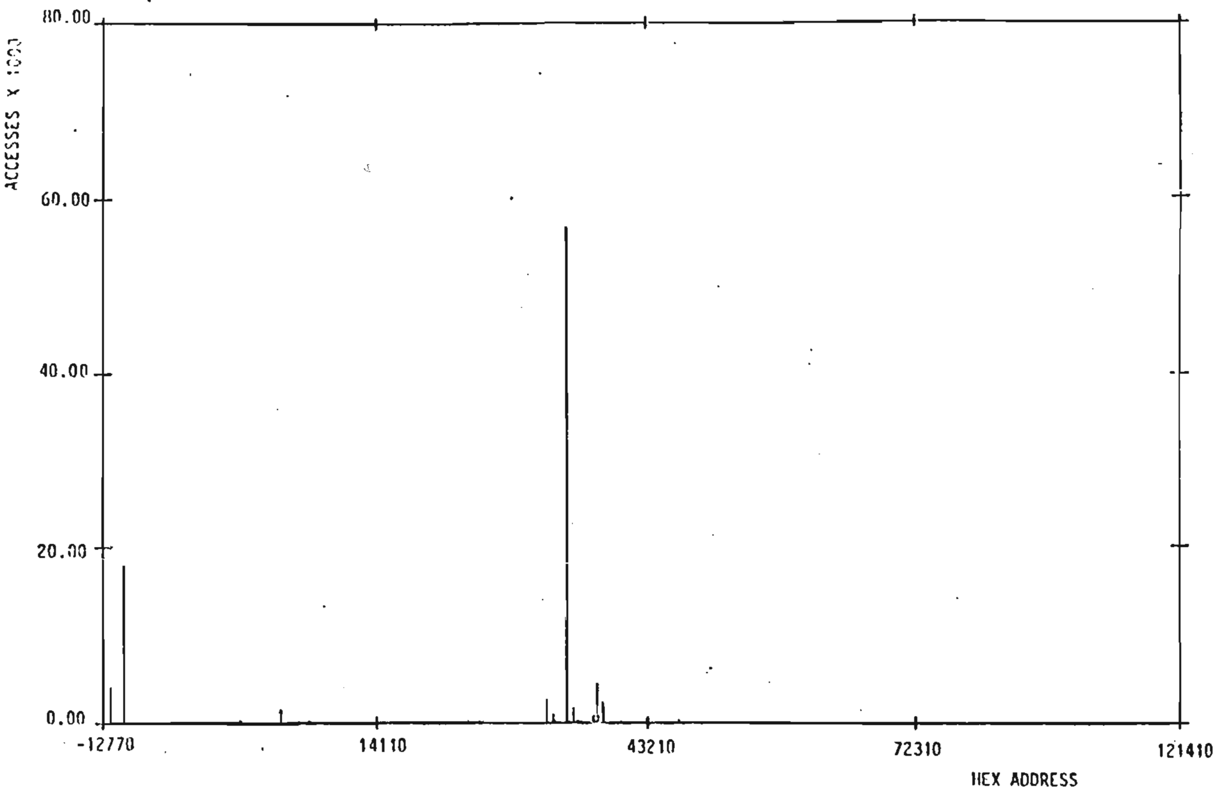
Load map of program CRNCH

```
          Address

          Begin  End
CRNCH     32042 37764
FEROR     37765 40027

OPEN      40030 40410  92067-16125 REV.2101 810615
CLOSE     40411 40625  92067-16125 REV.2101 801014
READF     40626 42135  92067-16125 REV.2101 810616
LURQ      42136 42550  92067-1X270 REV.2013 791024
PAUSE     42551 42650  24998-1X253 REV.2101 801007
LUTRU     42651 42757  92067-1X308 REV.2013 790223
OVRD.     42760 42760  92067-16125 REV.1903 780526
.DADS     42761 43070  24998-1X036 REV.2001 780818
.DMP      43071 43236  24998-1X045 REV.2001 780818
RMPAR     43237 43303  92068-1X025 REV.2101 800919
.DDI      43304 43604  24998-1X040 REV.2001 781021
SESSN     43605 43622  92067-16125 REV.1903 780413
R/W$      43623 43761  92067-16125 REV.2101 801013
.DNG      43762 43771  24998-1X046 REV.2001 780818
.DIO.     43772 44053  24998-1X331 REV.2101 800929
.EIO.     44054 45270  24998-1X329 REV.2101 800929
.FMCV     45271 47533  24998-1X333 REV.2101 800709
FMTIO     47534 50765  24998-1X328 REV.2101 800929
.IOER     50766 51101  24998-1X321 REV.2101 800731
.UFMP     51102 51114  24998-1X296 REV.2101 800731
PAU.E     51115 51115  24998-1X254 REV.2001 750701
PNAME     51116 51166  92068-1X035 REV.2101 800919
.RTOI     51167 51301  24998-1X063 REV.2013 791230
.FPWR     51302 51343  24998-1X124 REV.2001 781106
ALOG      51344 51456  24998-1X162 REV.2001 780424
.DDE      51457 51470  24998-1X039 REV.2001 780818
.DIN      51471 51476  24998-1X042 REV.2001 780818
ERR0      51477 51566  24998-1X250 REV.2001 771122
.SNCS     51567 51730  24998-1X159 REV.2001 780424
.CMRS     51731 52014  24998-1X171 REV.2001 780424
IFTTY     52015 52102  92067-1X295 REV.2013 790118
$ALRN     52103 52220  92067-1X271 REV.2013 770715
$OPEN     52221 52375  92067-16125 REV.1903 790103
RW$UB     52376 52747  92067-16125 REV.2101 800303
RWND$     52750 53077  92067-16125 REV.2101 810617
REIO      53100 53224  92067-1X275 REV.2013 790316
$SMVE     53225 53317  92067-1X483 REV.2013 800129
ERO.E     53320 53320  24998-1X249 REV.2001 750701
.OPN?     53321 53344  24998-1X325 REV.2101 800803
..FCM     53345 53361  24998-1X182 REV.2001 750701
```

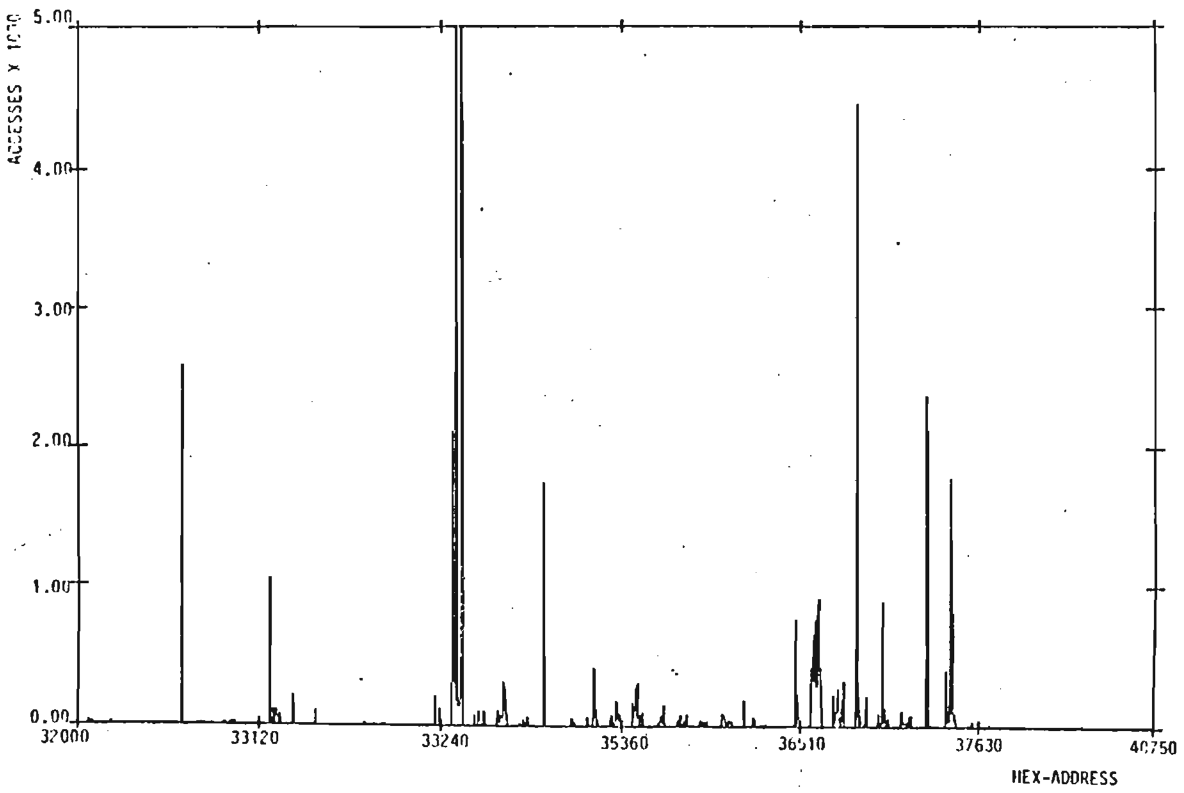
Listing of program CRNCH

```
FTN4,L
      PROGRAM CRNCH,3,99
      DIMENSION INAM1(3),IDCB1(144),BUF1(64)
      DIMENSION BUFA(1200)
      DATA INAM1/2H'F,2HIL,2HET/
      LU=1
      TNUM=0.0
1     CALL OPEN(IDCB1,IERR,INAM1,0,0,-52)
      CALL FEROR(IERR)
      DO 9 IC=1,1100
      BUFA(IC)=0.0
9     CONTINUE
      IM=0
25    CALL READF(IDCB1,IERR,BUF1,128,IL)
      CALL FEROR(IERR)
      IF (IL.EQ.-1) GO TO 27
      DO 26 IS=1,IL/2
      BUFA(IM+IS)=BUF1(IS)
      BUF1(IS)=0.0
26    CONTINUE
      IM=IM+IL/2
      GO TO 25
27    IN=0
      CALL CLOSE(IDCB1,IERR)
      CALL FEROR(IERR)
      DO 299 I=1,10000
      D=SIN(FLOAT(I))
      D=D**2
      E=ALOG(D)
      F=E**3
299   CONTINUE
      DO 301 I=1,10000
      D=SIN(FLOAT(I))
      D=D**2
      E=ALOG(D)
      F=E**3
301   CONTINUE
      CALL OPEN(IDCB1,IERR,INAM1,0,0,-52)
      CALL FEROR(IERR)
      TNUM=BUFA(1)
      TNUM = TNUM + 1.0
      BUFA(1)=TNUM
30    DO 31 IX=1,50
      BUF1(IX)=BUFA(IX+IN)
      IF(IN.GT.IM) GO TO 32
31    CONTINUE
      IN=IN+50
      CALL WRITF(IDCB1,IERR,BUF1,100)
      CALL FEROR(IERR)
      GO TO 30
```

```
 32     CALL CLOSE(IDCB1,IERR)
        CALL FEROR(IERR)
        WRITE(LU,150)TNUM
 150    FORMAT( " COMPLETED THE ",F8.1," 'TH RUN "/
      ?" ANOTHER IN 20 SECONDS. ")
        CALL EXEC(12,0,2,0,-20)
        IF(TNUM.GE.4.0) GO TO 99
        GO TO 1
 99     STOP
        END
```



Histogram of operation of program RESP



Detail of program RESP histogram

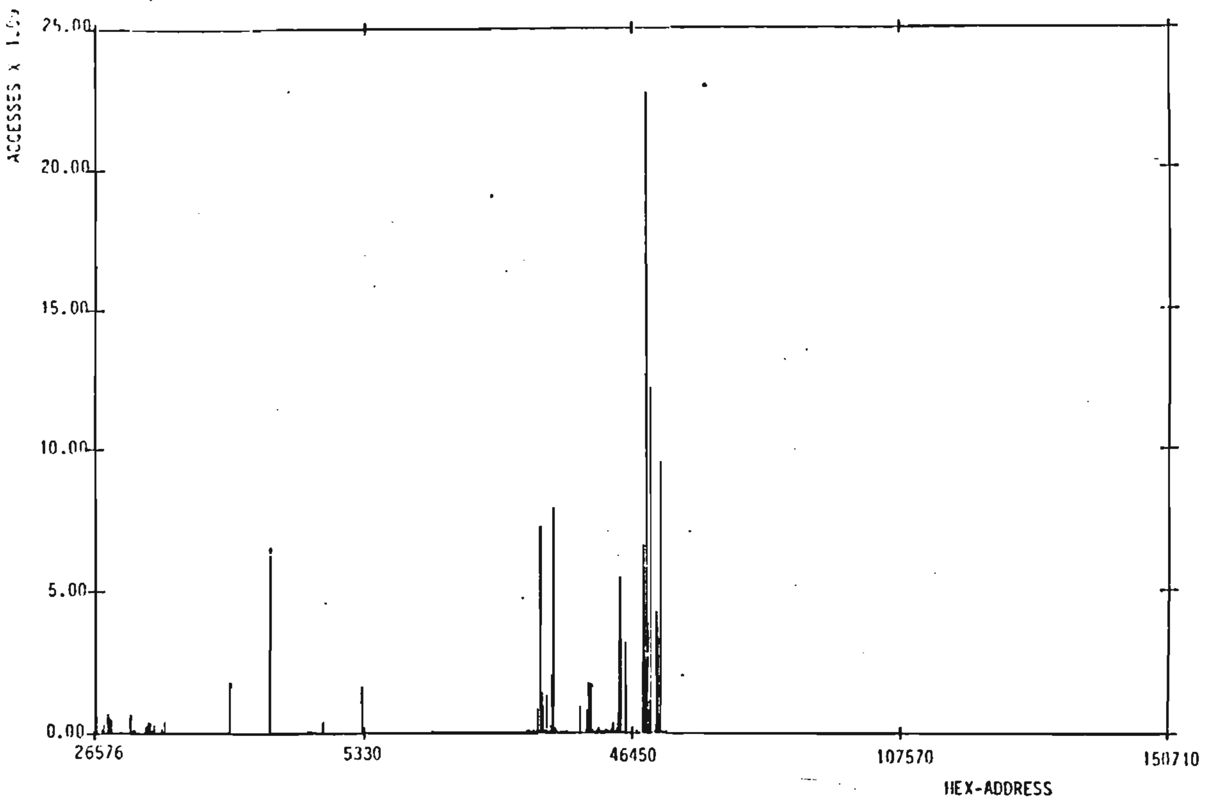
Load map of program RESP

Address

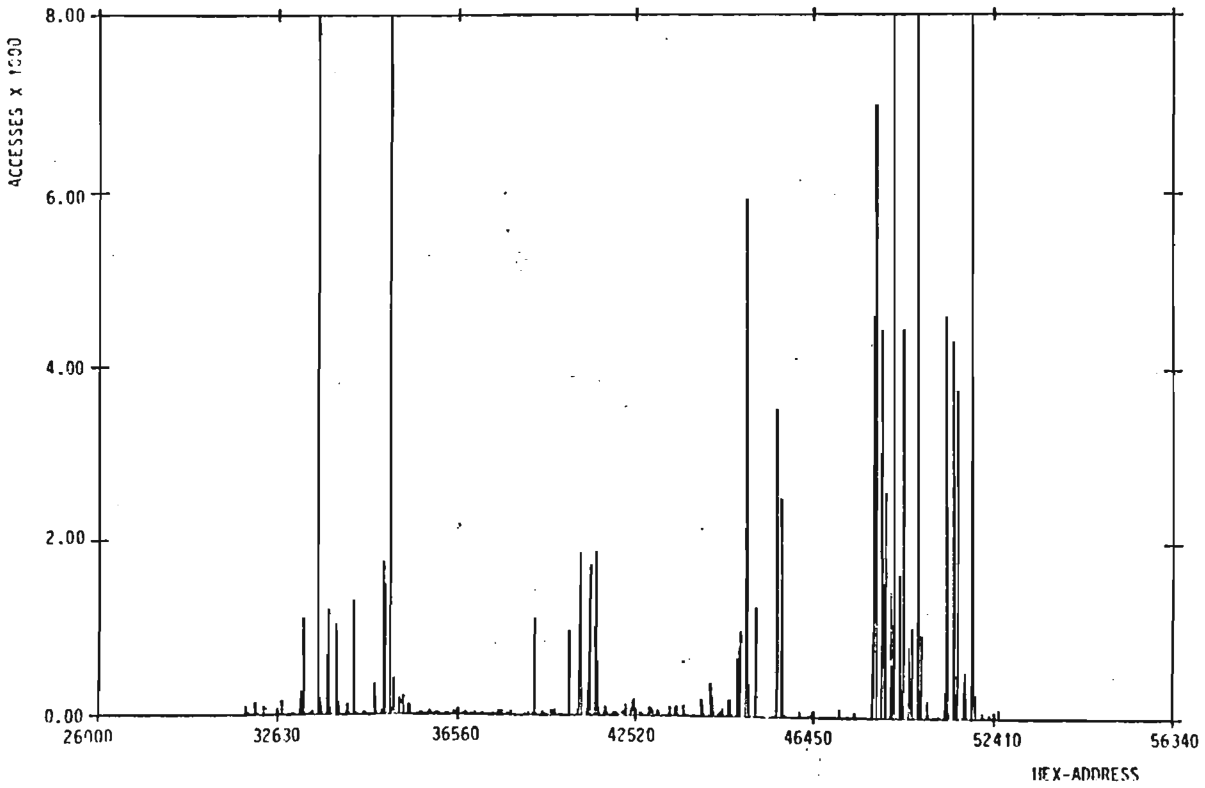
```
            Begin  End

RESP        32042 32142

PAUSE       32143 32242   24998-1X253 REV.2101 801007
.DIO.       32243 32324   24998-1X331 REV.2101 800929
.EIO.       32325 33541   24998-1X329 REV.2101 800929
.FMCV       33542 36004   24998-1X333 REV.2101 800709
FMTIO       36005 37236   24998-1X328 REV.2101 800929
.IOER       37237 37352   24998-1X321 REV.2101 800731
.UFMP       37353 37365   24998-1X296 REV.2101 800731
PAU.E       37366 37366   24998-1X254 REV.2001 750701
PNAME       37367 37437   92068-1X035 REV.2101 800919
REIO        37440 37564   92067-1X275 REV.2013 790316
ERO.E       37565 37565   24998-1X249 REV.2001 750701
.OPN?       37566 37611   24998-1X325 REV.2101 800803
```

Listing of program RESP

```
FTN4,L,M
      PROGRAM RESP,3,99
      NUM=0
1     NUM=NUM+1
      WRITE(1,100)NUM
100   FORMAT(I8)
      DO 20 I=1,1000
      A=FLOAT(I)
20    CONTINUE
      WRITE(1,100)NUM
      CALL EXEC(12,0,1,0,-100)
      IF(NUM.GT.200)GO TO 99
      GO TO 1
99    STOP
      END
```



Histogram of operation of program FILE



Detail of program FILE histogram

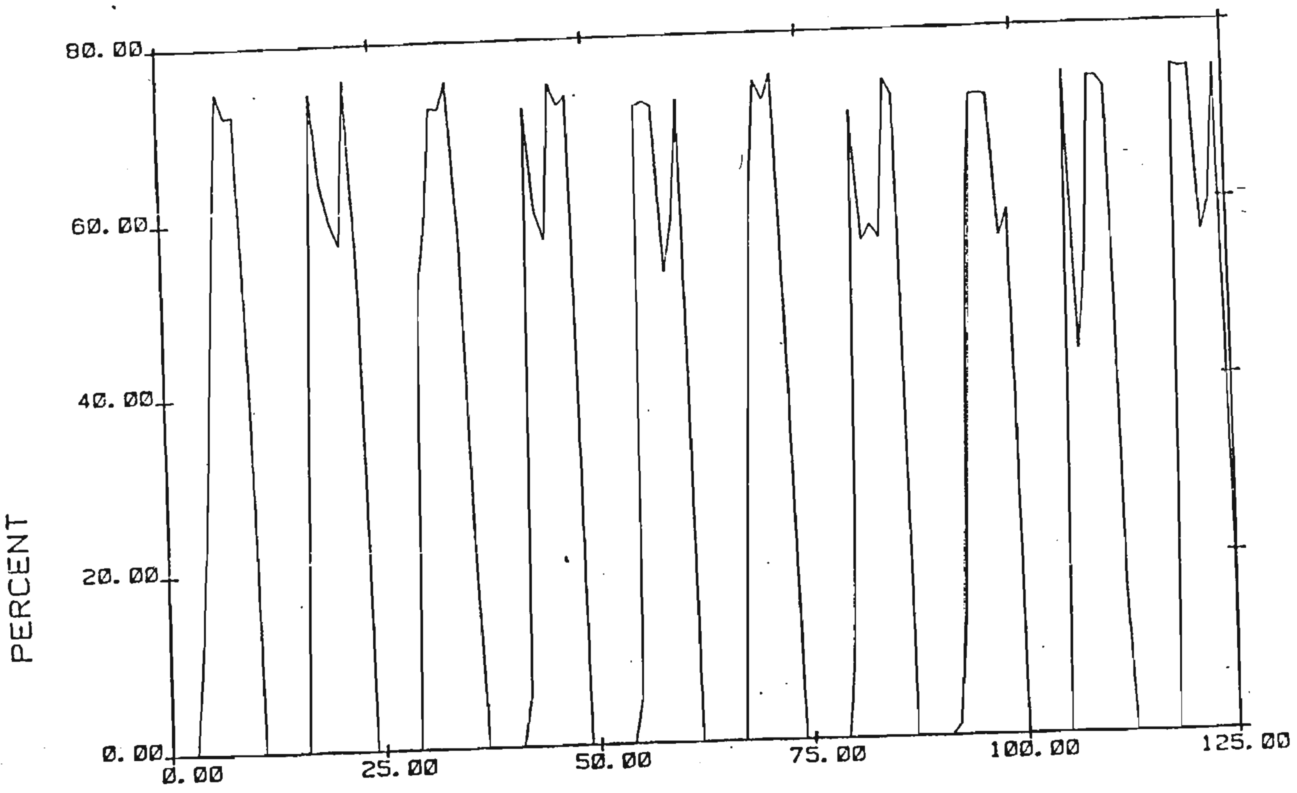
Load map of program FILE

```
        Address

        Begin  End

FILE    32042 37705
FEROR   37706 37750

OPEN    37751 40331  92067-16125 REV.2101 810615
CLOSE   40332 40546  92067-16125 REV.2101 801014
READF   40547 42056  92067-16125 REV.2101 810616
LURQ    42057 42471  92067-1X270 REV.2013 791024
PAUSE   42472 42571  24998-1X253 REV.2101 801007
LUTRU   42572 42700  92067-1X308 REV.2013 790223
OVRD.   42701 42701  92067-16125 REV.1903 780526
.DADS   42702 43011  24998-1X036 REV.2001 780818
.DMP    43012 43157  24998-1X045 REV.2001 780818
RMPAR   43160 43224  92068-1X025 REV.2101 800919
.DDI    43225 43525  24998-1X040 REV.2001 781021
SESSN   43526 43543  92067-16125 REV.1903 780413
R/W$    43544 43702  92067-16125 REV.2101 801013
.DNG    43703 43712  24998-1X046 REV.2001 780818
.DIO.   43713 43774  24998-1X331 REV.2101 800929
.EIO.   43775 45211  24998-1X329 REV.2101 800929
.FMCV   45212 47454  24998-1X333 REV.2101 800709
FMTIO   47455 50706  24998-1X328 REV.2101 800929
.IOER   50707 51022  24998-1X321 REV.2101 800731
.UFMP   51023 51035  24998-1X296 REV.2101 800731
PAU.E   51036 51036  24998-1X254 REV.2001 750701
PNAME   51037 51107  92068-1X035 REV.2101 800919
.DDE    51110 51121  24998-1X039 REV.2001 780818
.DIN    51122 51127  24998-1X042 REV.2001 780818
IFTTY   51130 51215  92067-1X295 REV.2013 790118
$ALRN   51216 51333  92067-1X271 REV.2013 770715
$OPEN   51334 51510  92067-16125 REV.1903 790103
RW$UB   51511 52062  92067-16125 REV.2101 800303
RWND$   52063 52212  92067-16125 REV.2101 810617
REIO    52213 52337  92067-1X275 REV.2013 790316
$SMVE   52340 52432  92067-1X483 REV.2013 800129
ERO.E   52433 52433  24998-1X249 REV.2001 750701
.OPN?   52434 52457  24998-1X325 REV.2101 800803
```

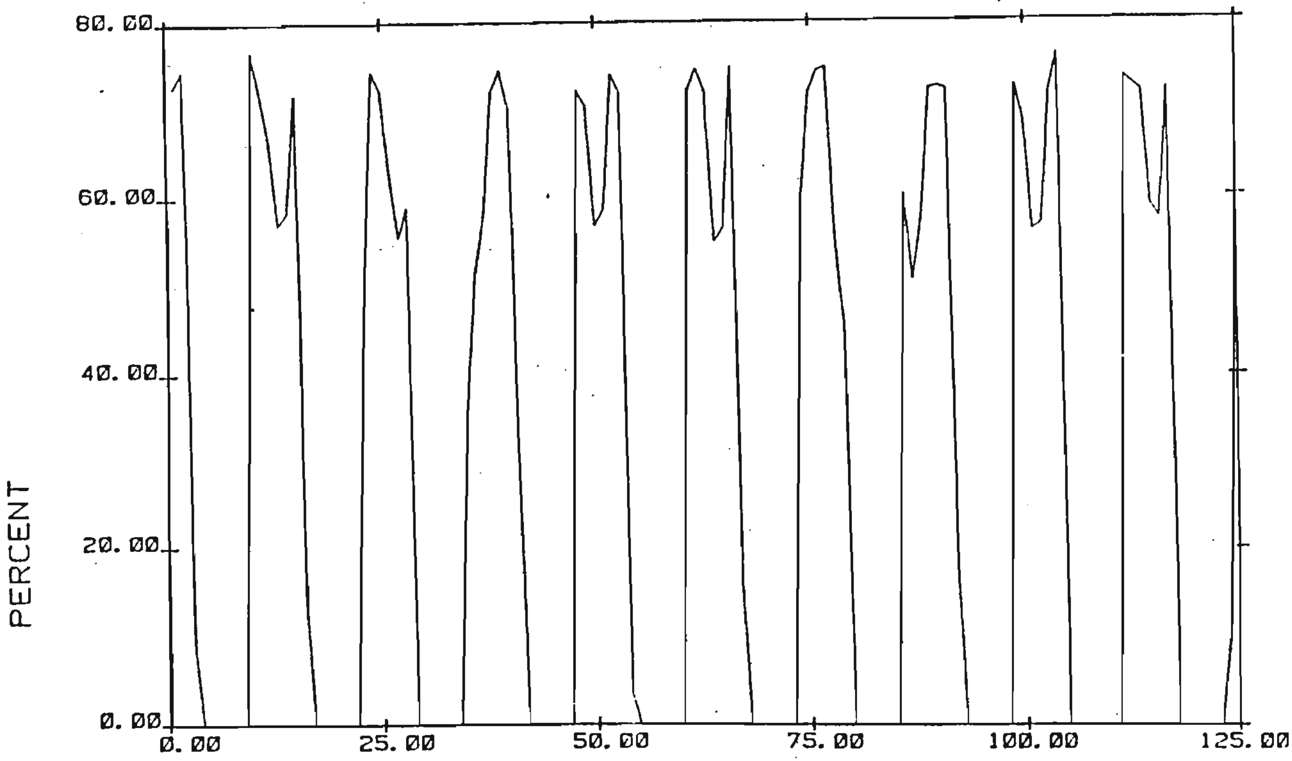
Listing of program FILE

```
FTN4,L,M
      PROGRAM FILE ,3,99
      DIMENSION INAM1(3),IDCB1(144),BUF1(64)
      DIMENSION BUFA(1200)
      DATA INAM1/2H'F,2HIL,2HER/
      LU=1
      TNUM=0.0
1     CALL OPEN(IDCB1,IERR,INAM1,0,0,-52)
      CALL FEROR(IERR)
      DO 9 IC=1,1100
      BUFA(IC)=0.0
9     CONTINUE
      IM=0
25    CALL READF(IDCB1,IERR,BUF1,128,IL)
      CALL FEROR(IERR)
      IF (IL.EQ.-1) GO TO 27
      DO 26 IS=1,IL/2
      BUFA(IM+IS)=BUF1(IS)
      BUF1(IS)=0.0
26    CONTINUE
      IM=IM+IL/2
      GO TO 25
27    IN=0
      CALL CLOSE(IDCB1,IERR)
      CALL FEROR(IERR)
      CALL OPEN(IDCB1,IERR,INAM1,0,0,-52)
      CALL FEROR(IERR)
      TNUM=BUFA(1)
      TNUM = TNUM + 1.0
      BUFA(1)=TNUM
30    DO 31 IX=1,50
      BUF1(IX)=BUFA(IX+IN)
      IF(IN.GT.IM) GO TO 32
31    CONTINUE
      IN=IN+50
      CALL WRITF(IDCB1,IERR,BUF1,100)
      CALL FEROR(IERR)
      GO TO 30
32    CALL CLOSE(IDCB1,IERR)
      CALL FEROR(IERR)
400   FORMAT( " ANOTHER ONE AND " )
      DO 277 IV = 1,200
      WRITE(1,400)
277   CONTINUE
      WRITE(LU,150)TNUM
150   FORMAT( " COMPLETED THE ",F8.1," 'TH RUN "/
     ?" ANOTHER IN 10 SECONDS. ")
      CALL EXEC(12,0,2,0,-10)
      IF(TNUM.GE.10.0) GO TO 99
      GO TO 1
99    STOP
      END
```

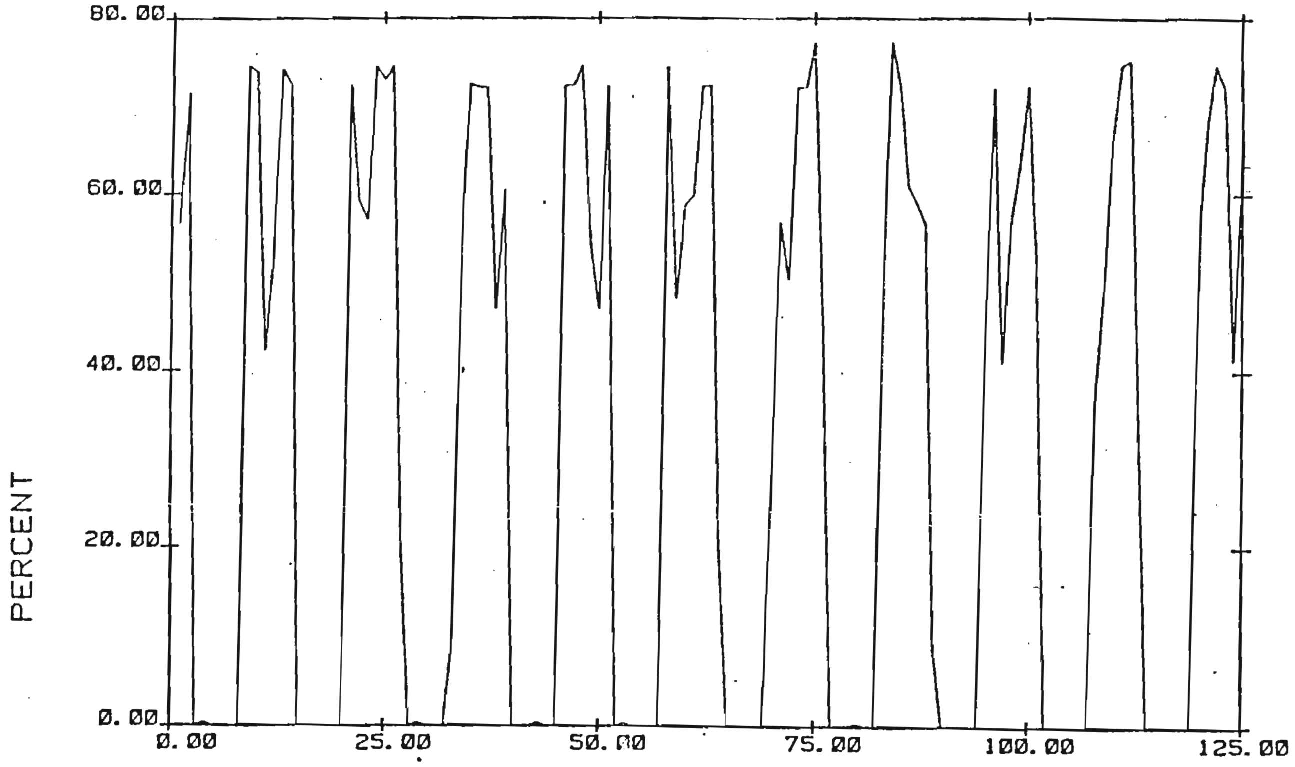
APPENDIX C : COMPLETE TEST RESULTS



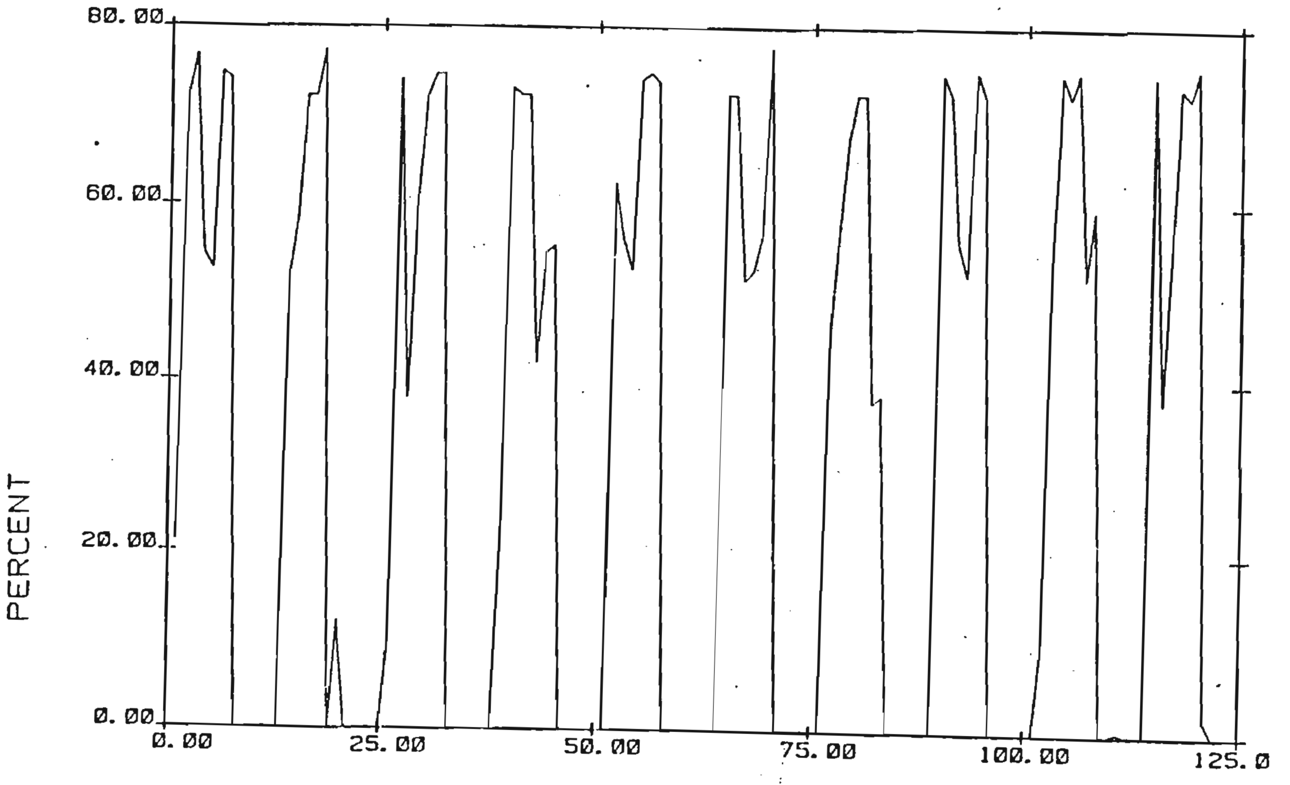
Utilization graph of IPP$_1$ - Test 1



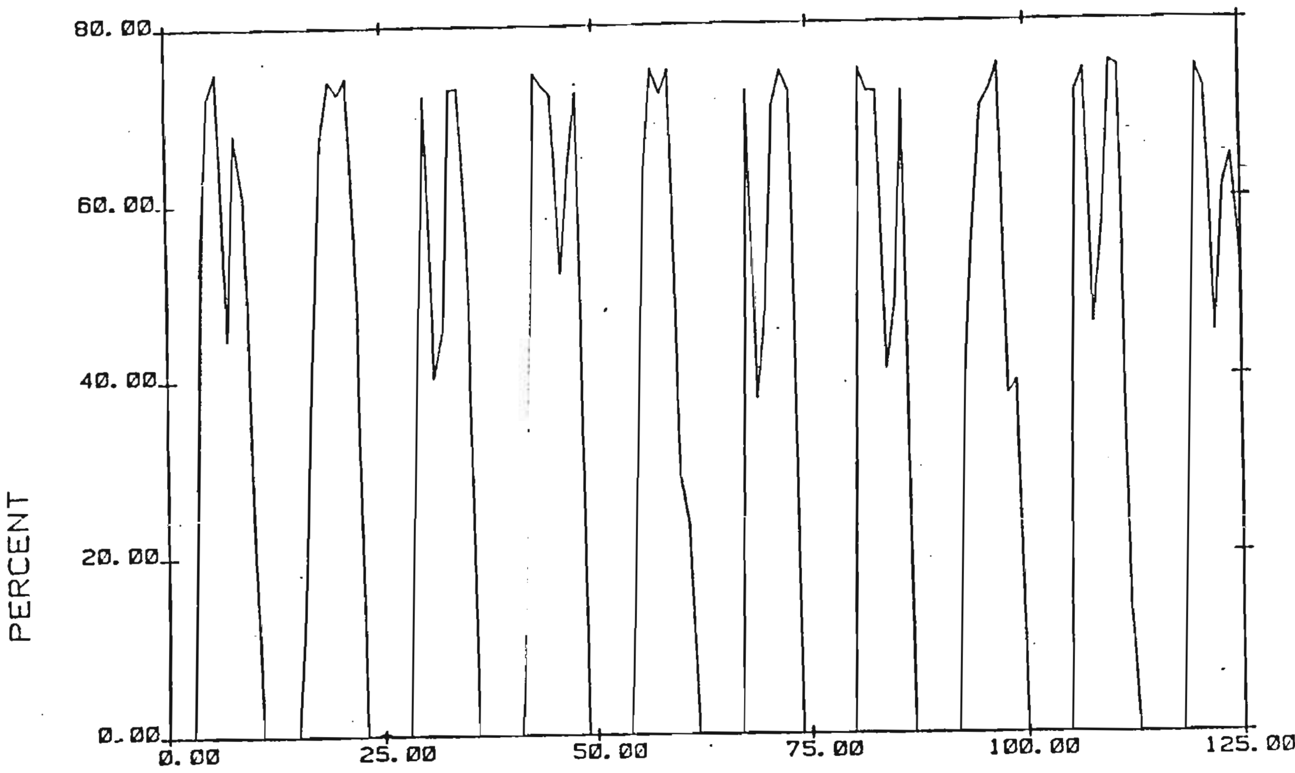
Utilization graph of IPP$_1$ - Test 2



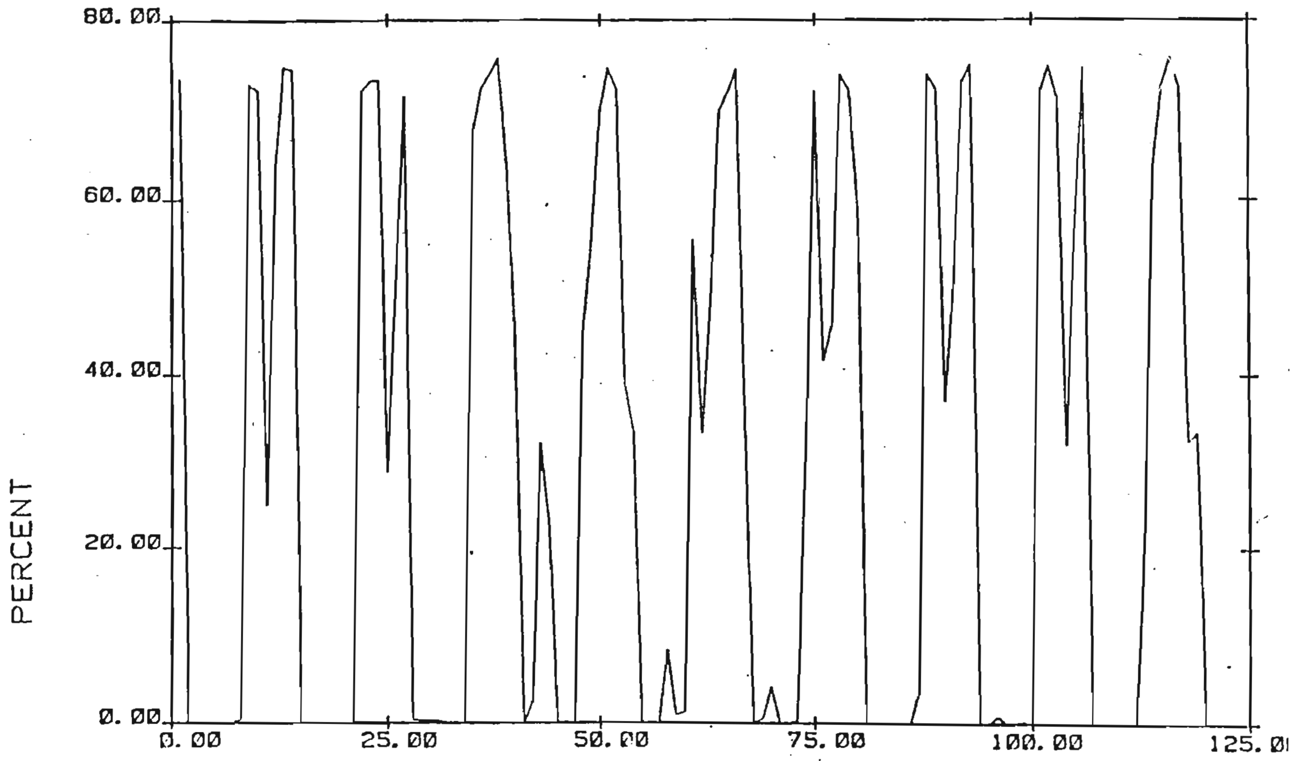
Utilization graph of IPP$_1$ - Test 3



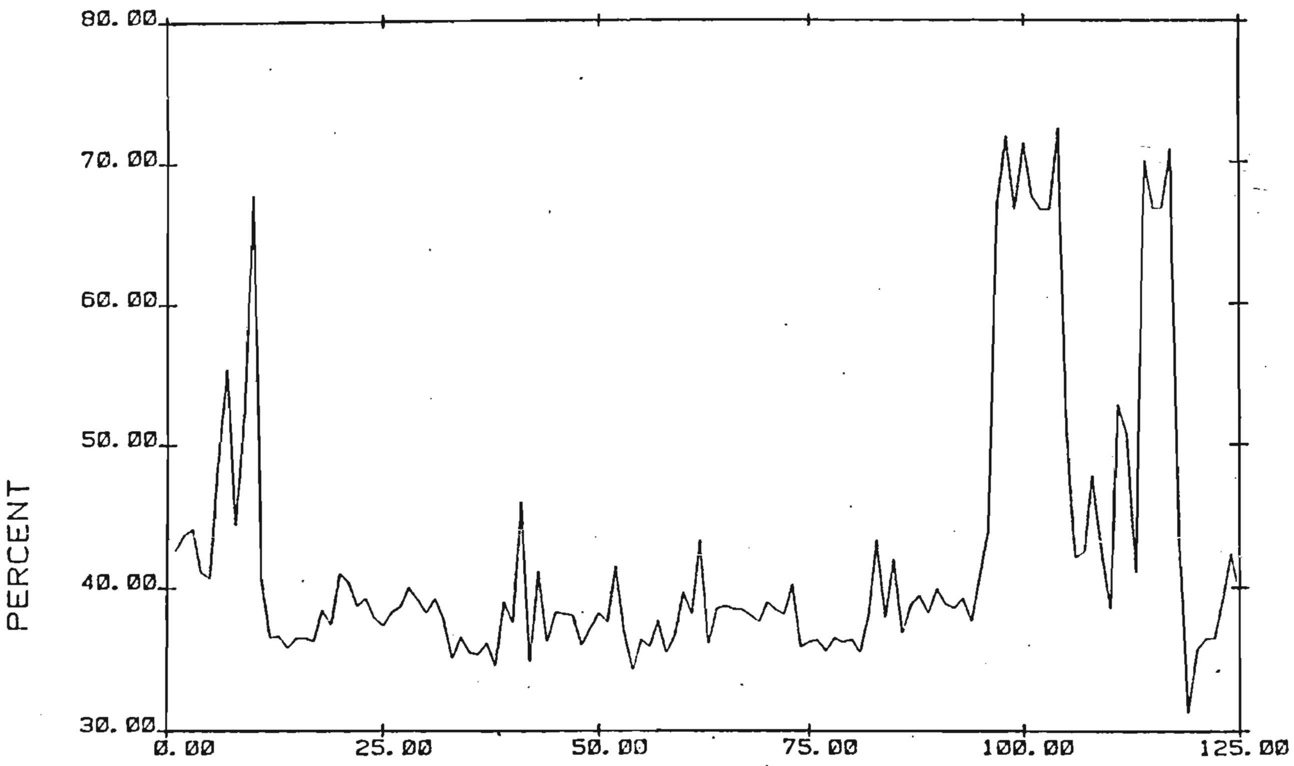
Utilization graph of IPP$_1$ - Test 4



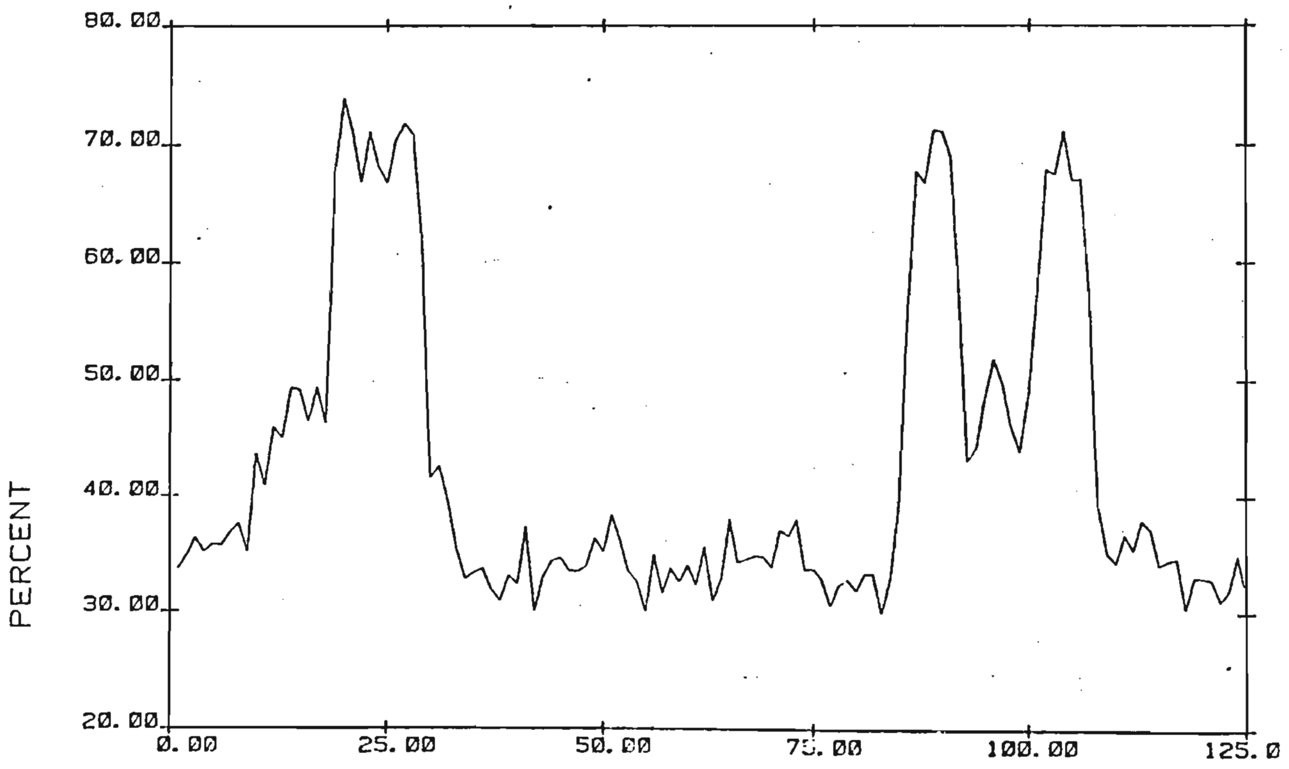
Utilization graph of IPP$_1$ - Test 5



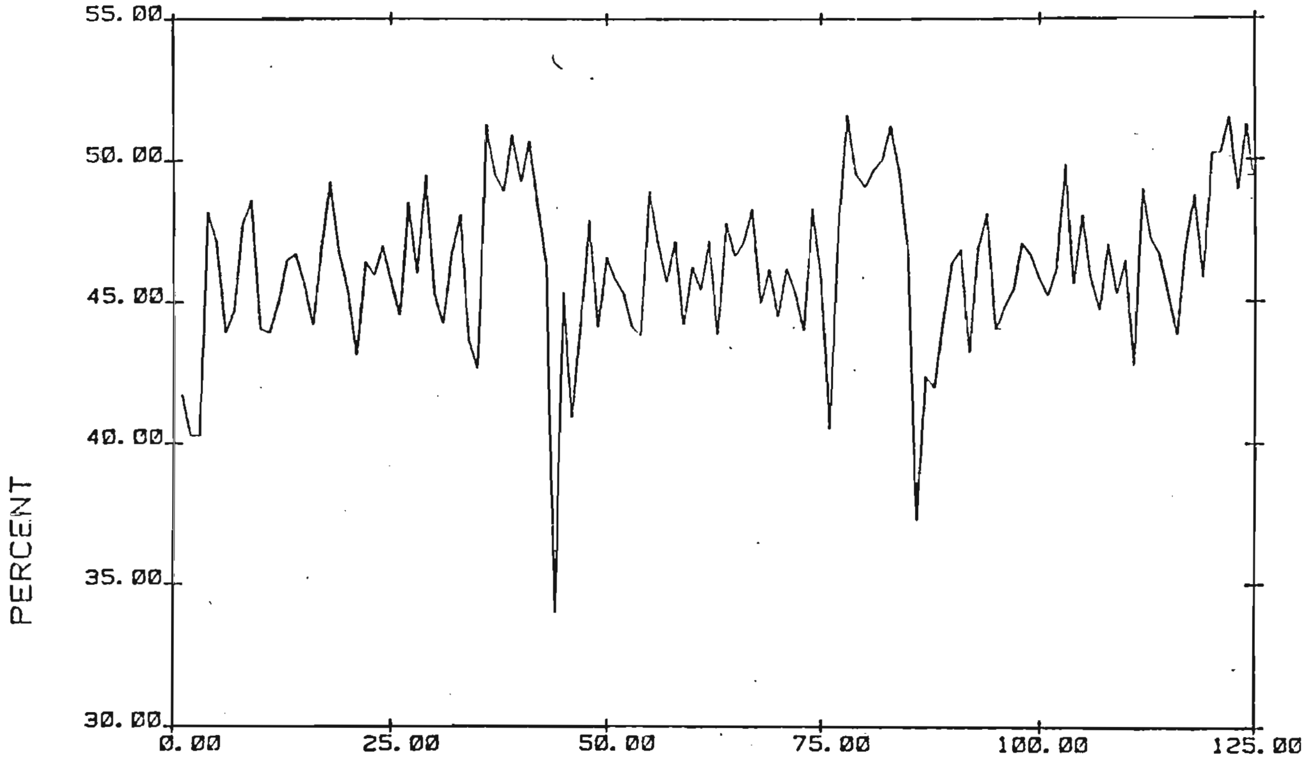
Utilization graph of IPP$_1$ - Test 6



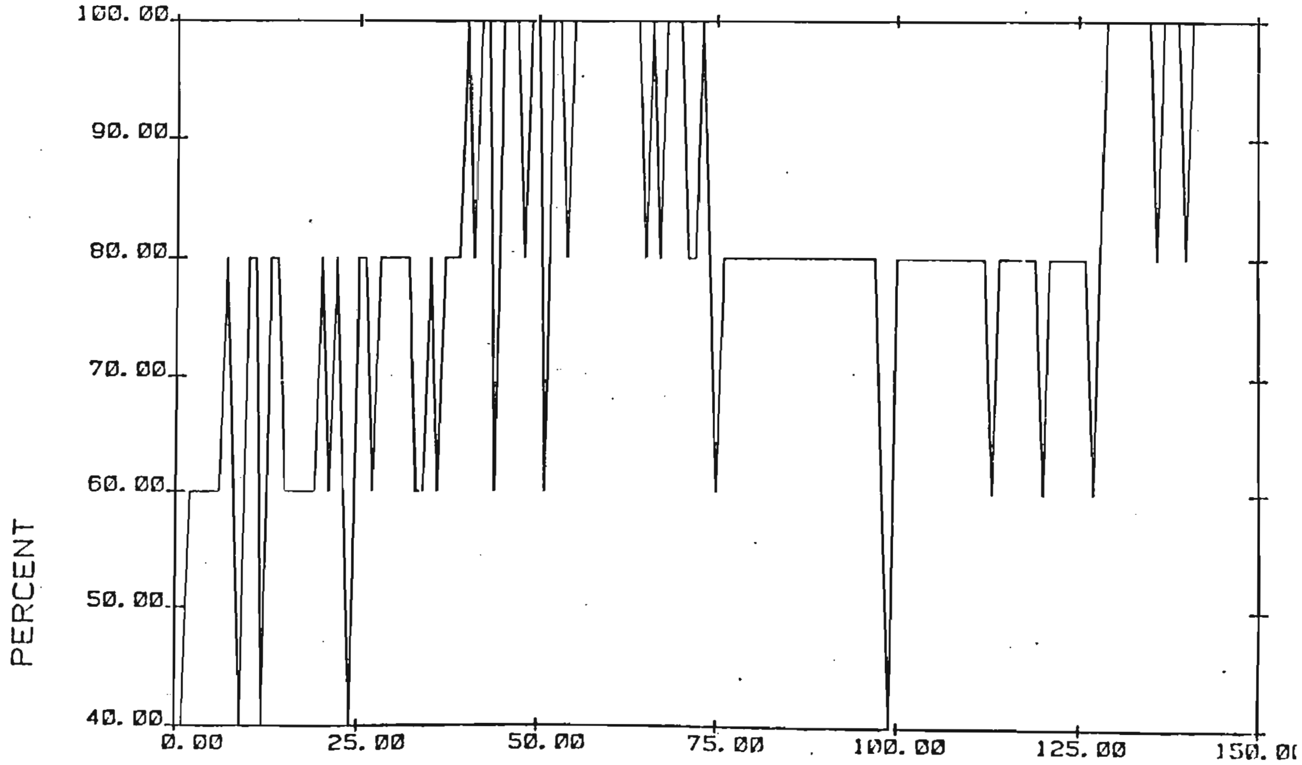
Utilization graph of IPP_1 - Test 7



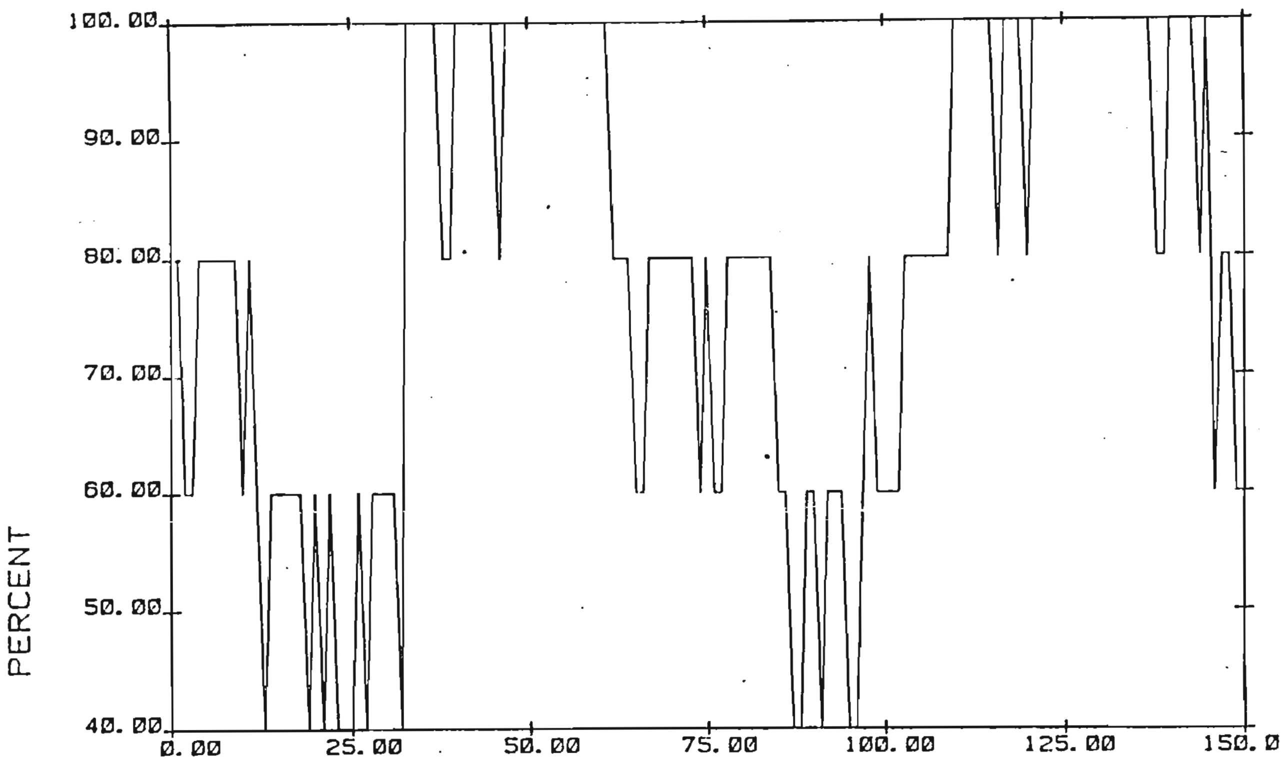
Utilization graph of IPP_1 - Test 8



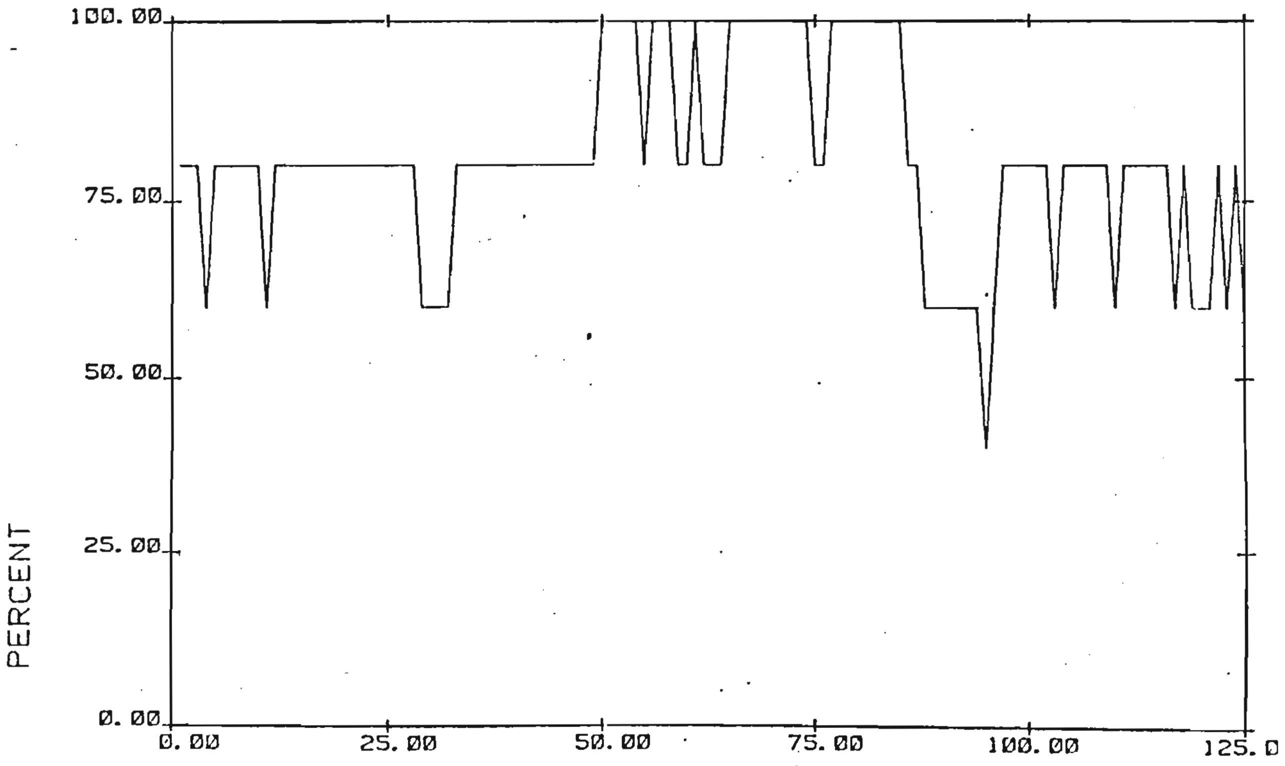
Utilization graph of IPP$_1$ - Test 9



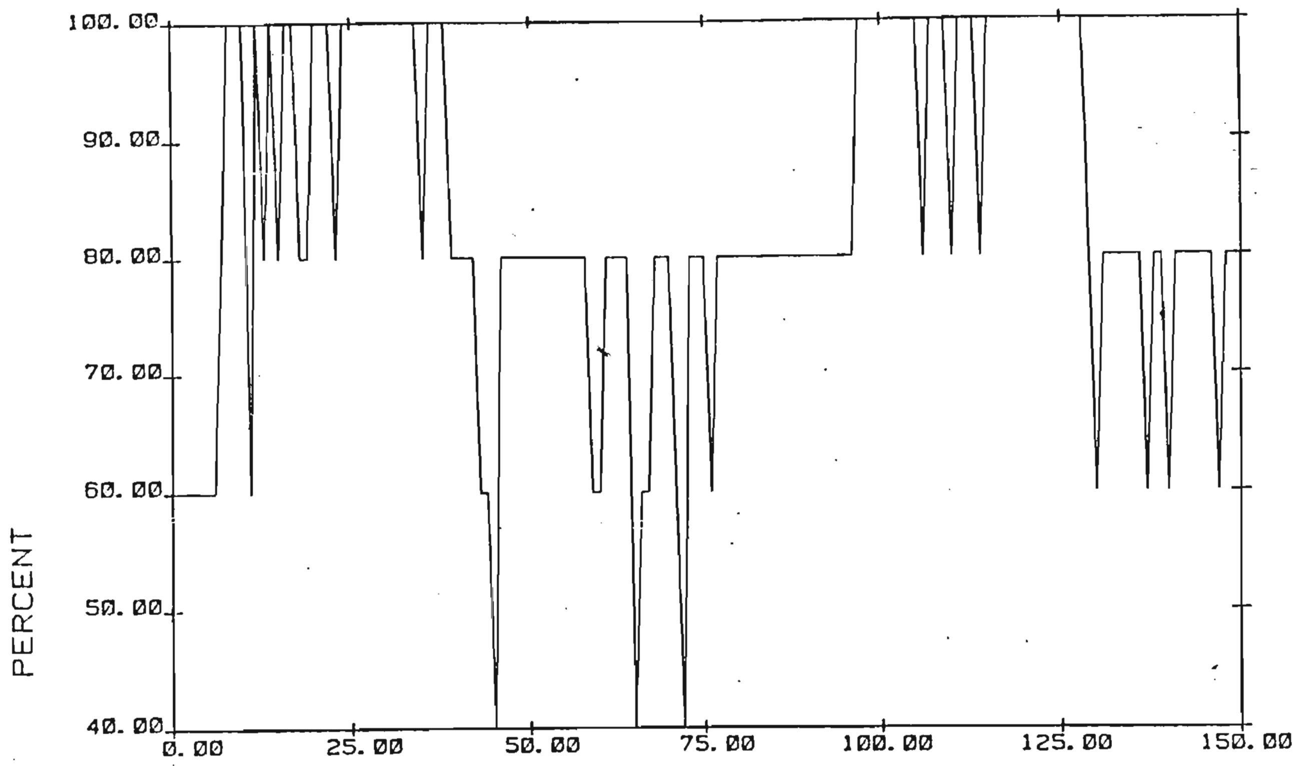
Utilization graph of ISP$_1$ - Test 1



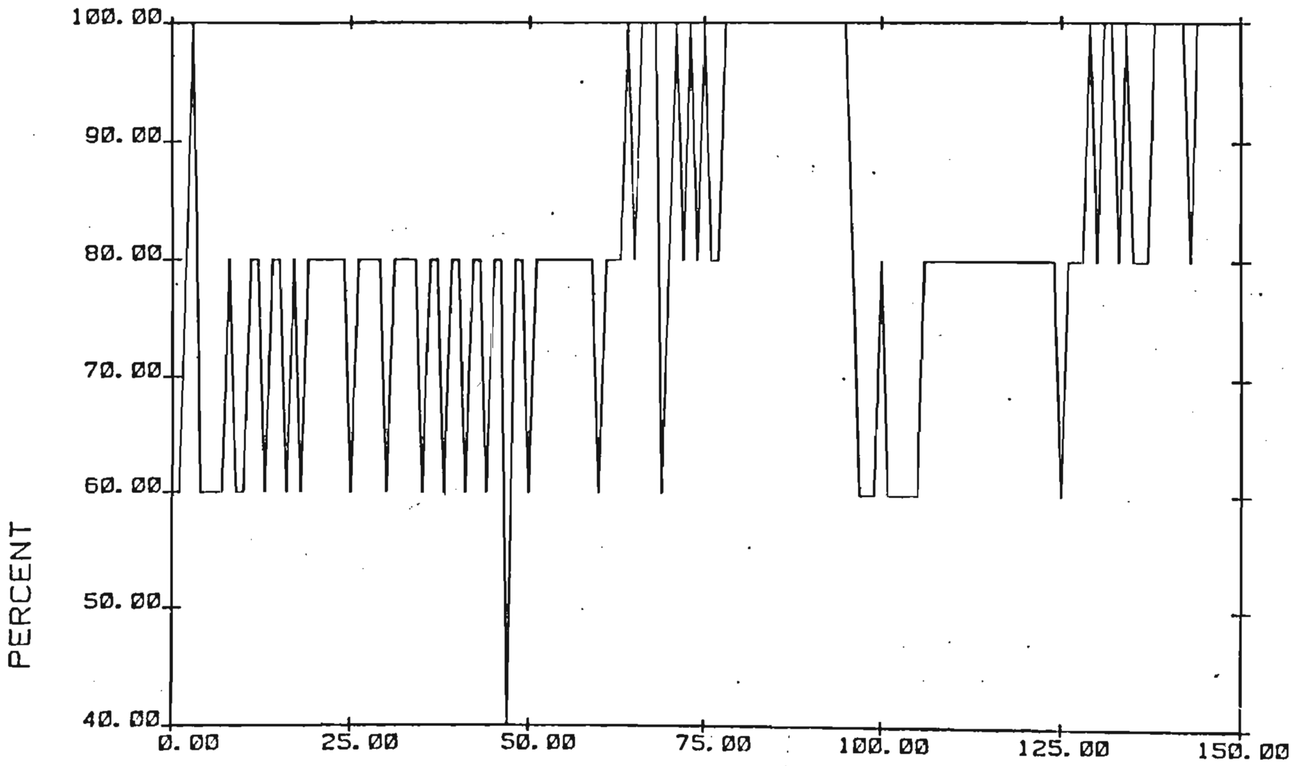
Utilization graph of ISP$_1$ - Test 2



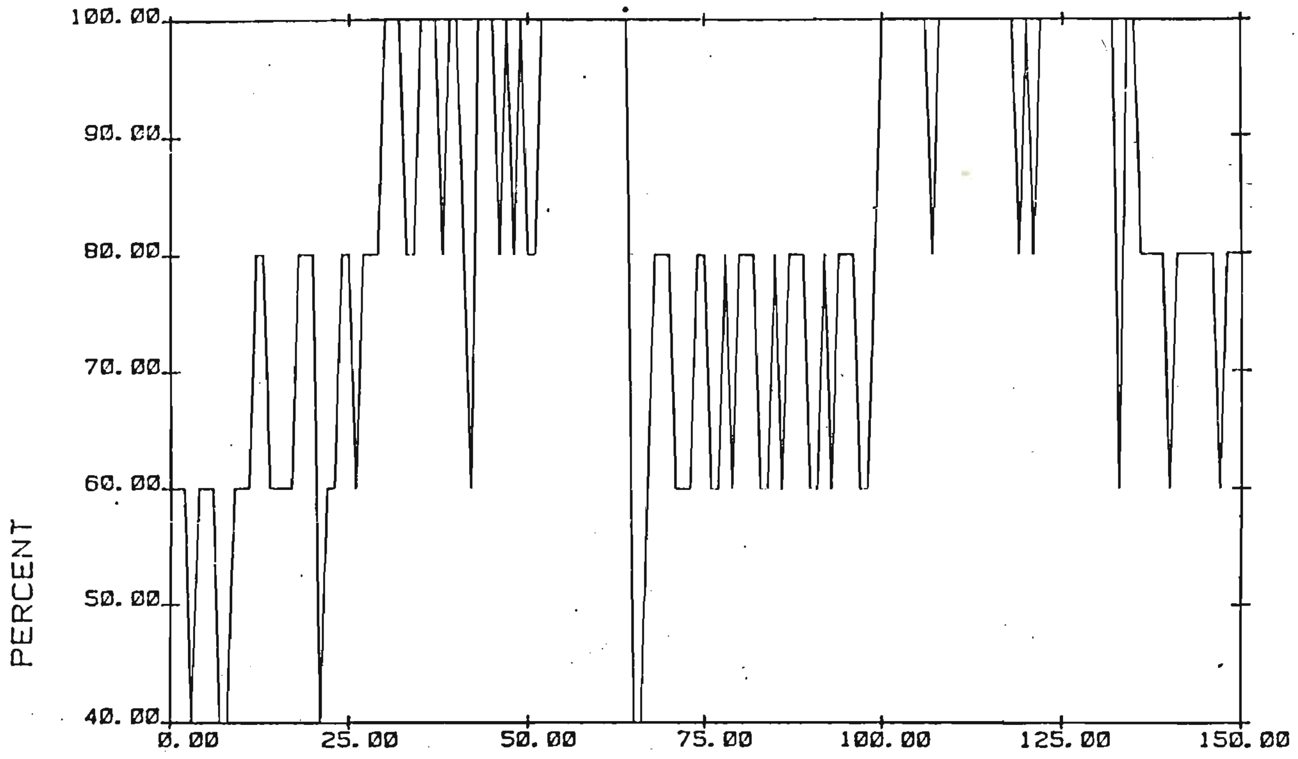
Utilization graph of ISP$_1$ - Test 3



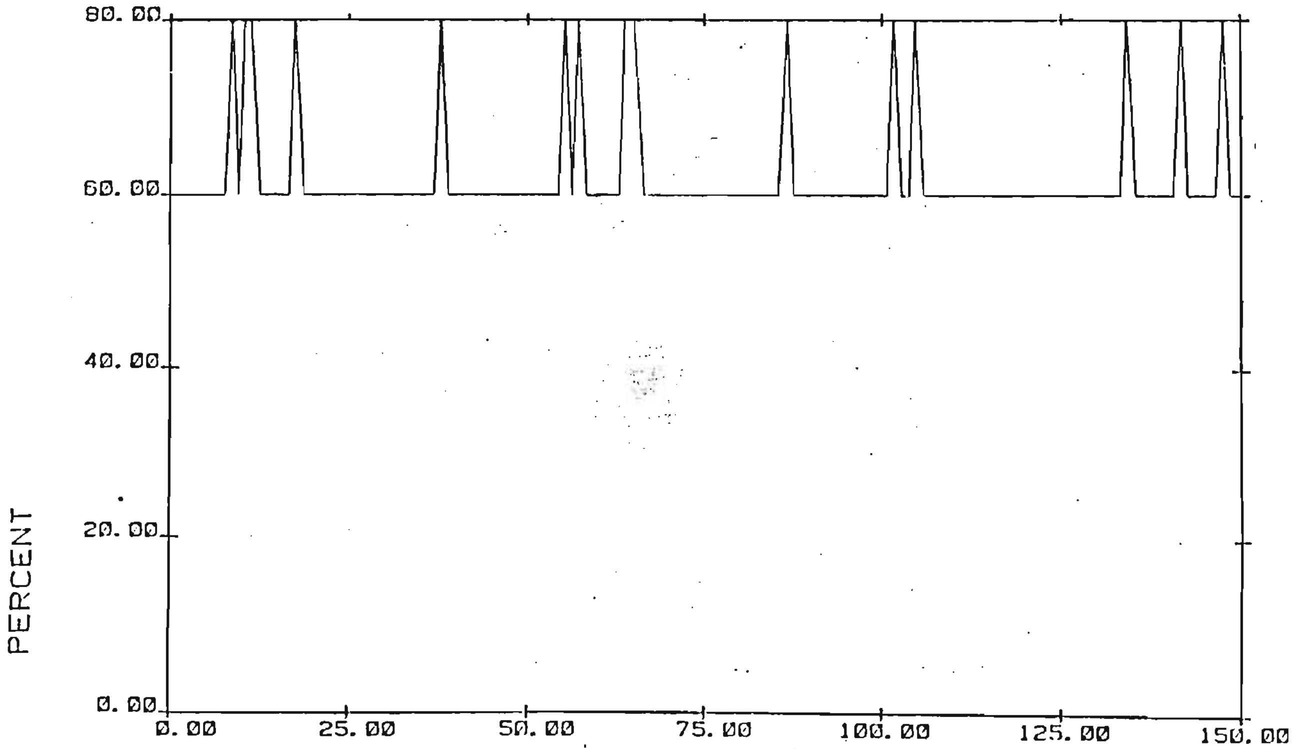
Utilization graph of ISP$_1$ - Test 4



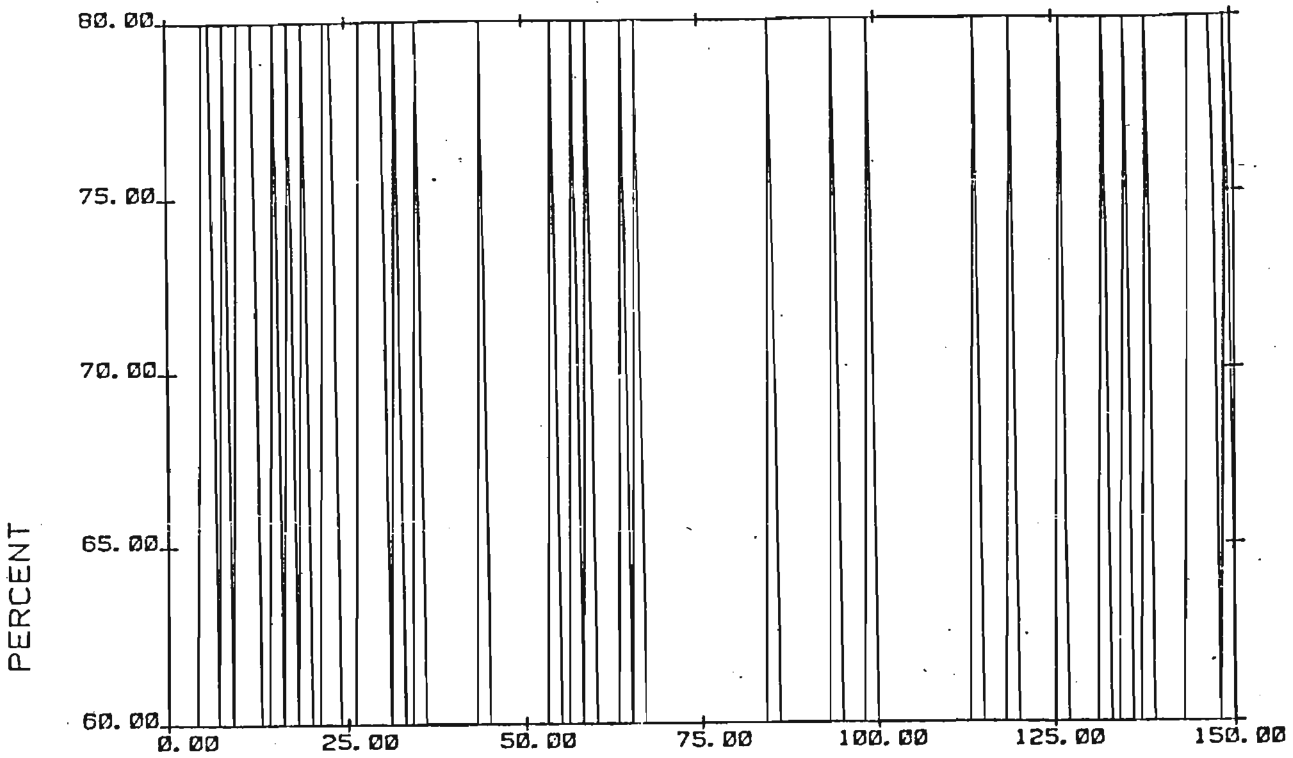
Utilization graph of ISP$_1$ - Test 5



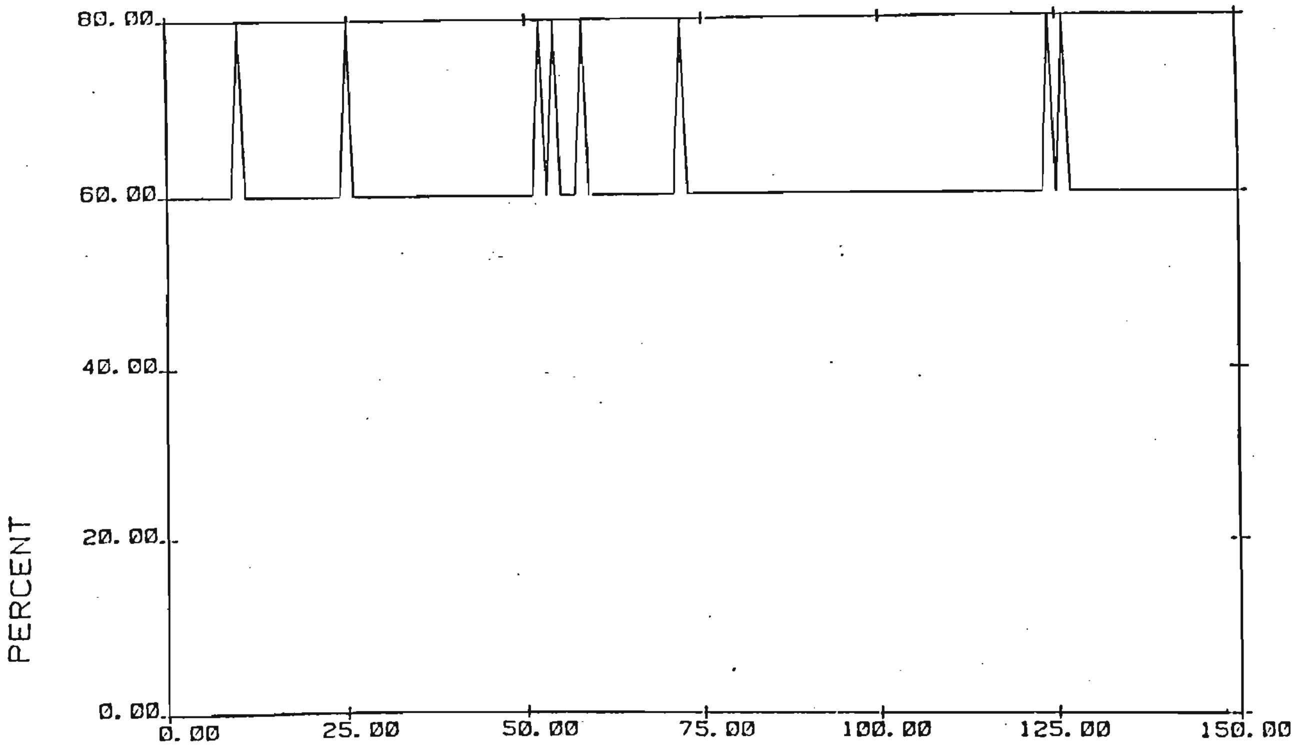
Utilization graph of ISP_1 - Test 6



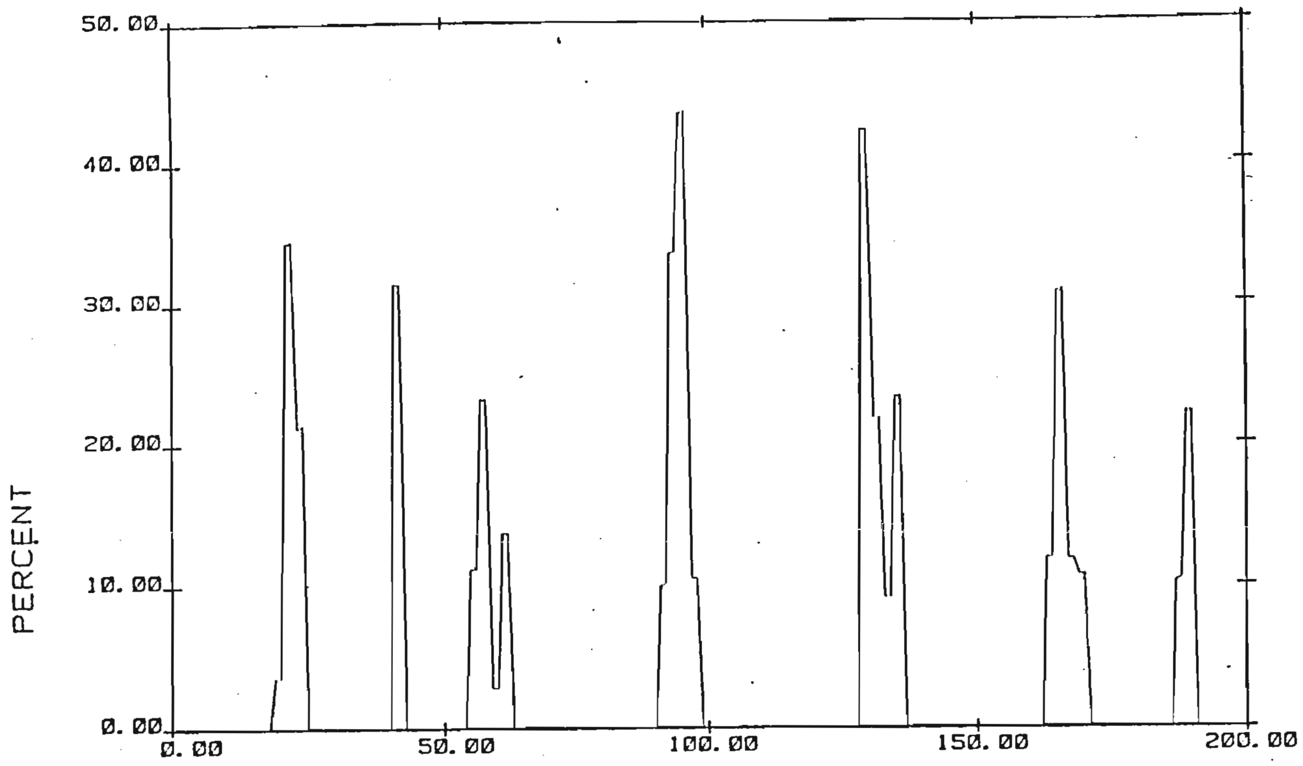
Utilization graph of ISP_1 - Test 7



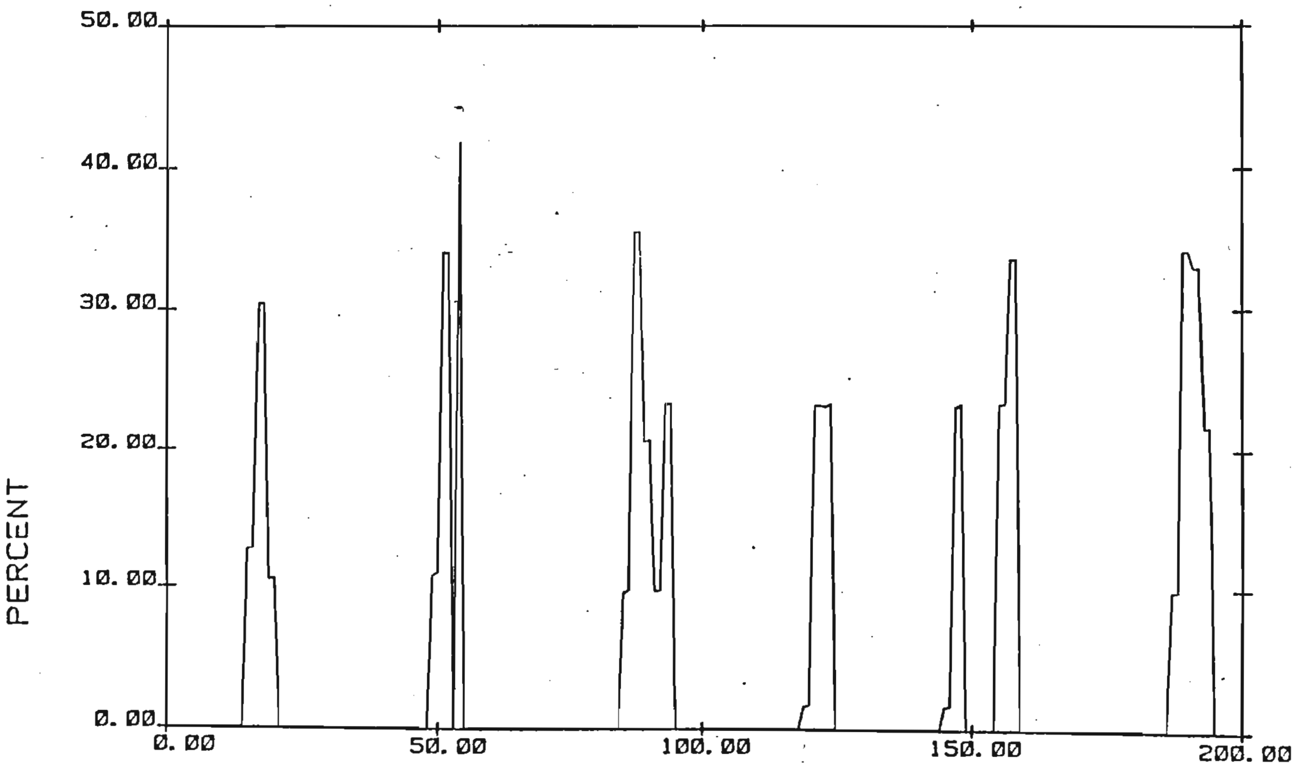
Utilization graph of ISP$_1$ - Test 8



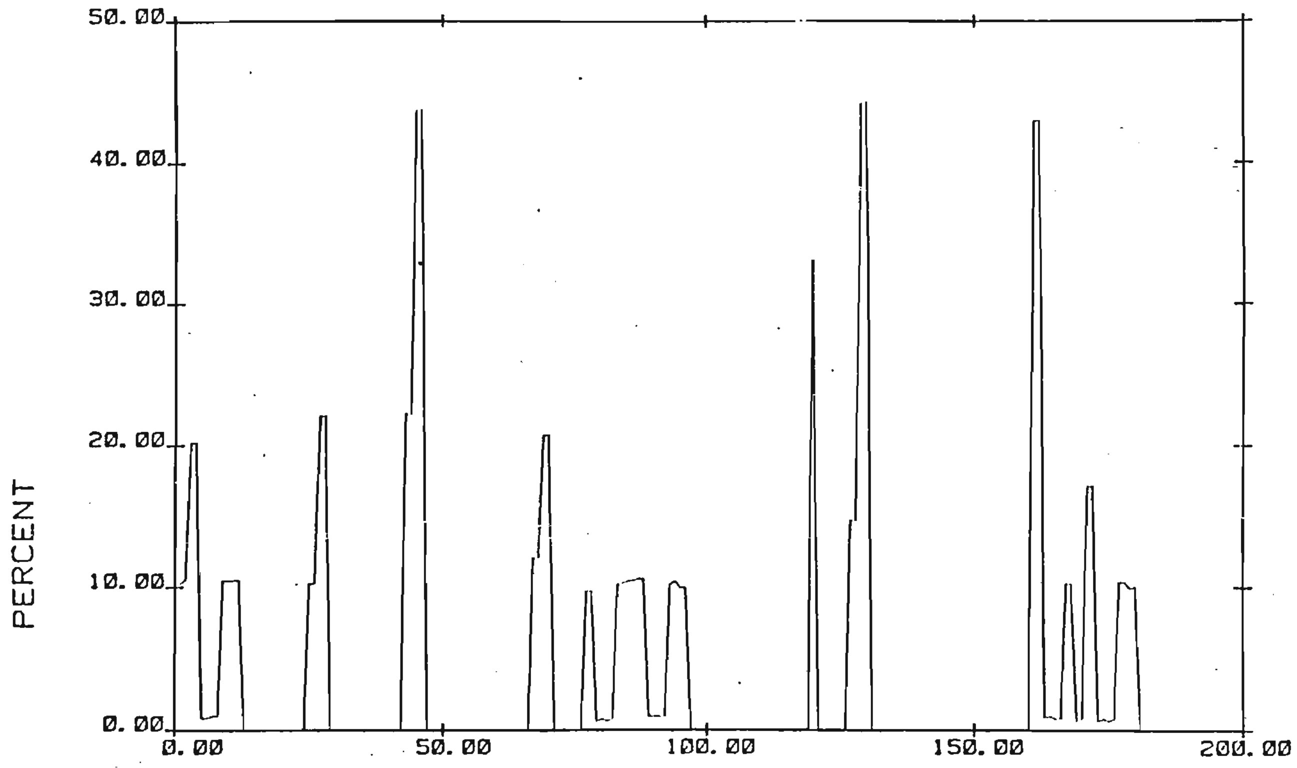
Utilization graph of ISP$_1$ - Test 9



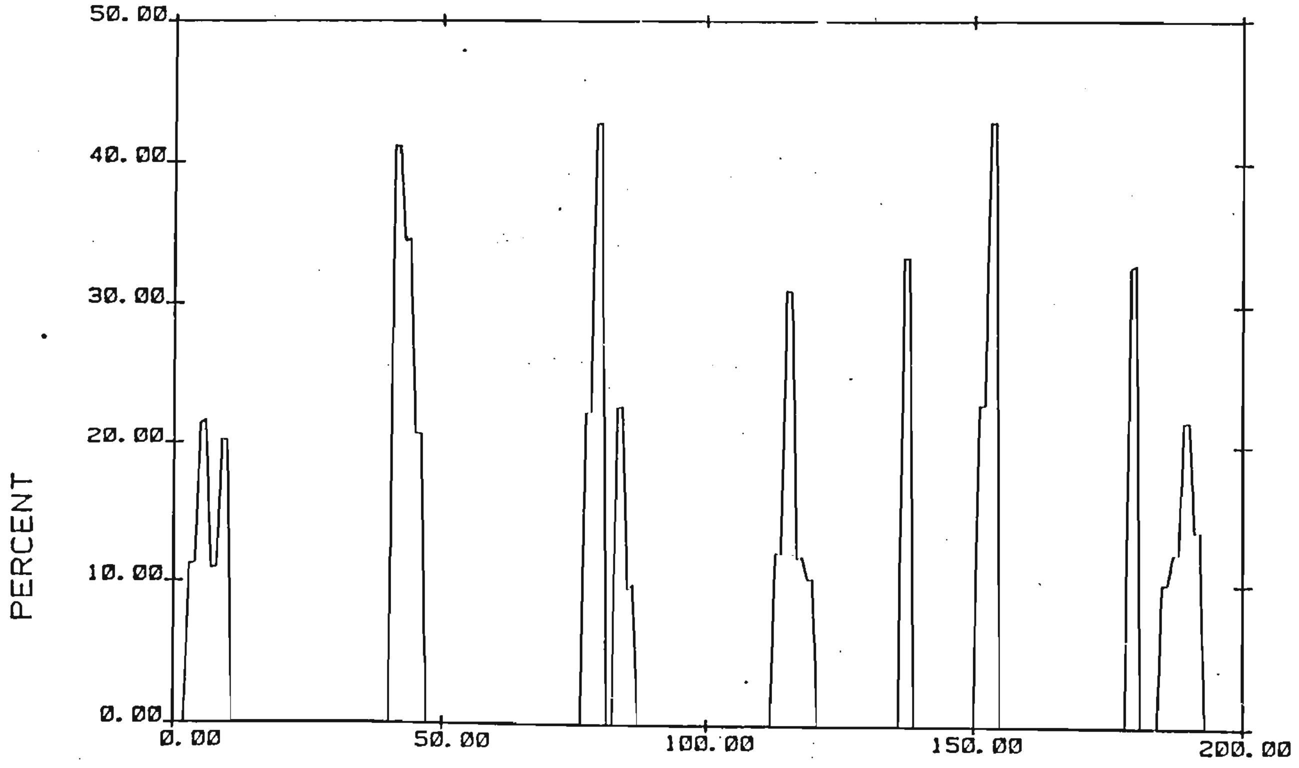
Utilization graph of ISP$_2$ - Test 1



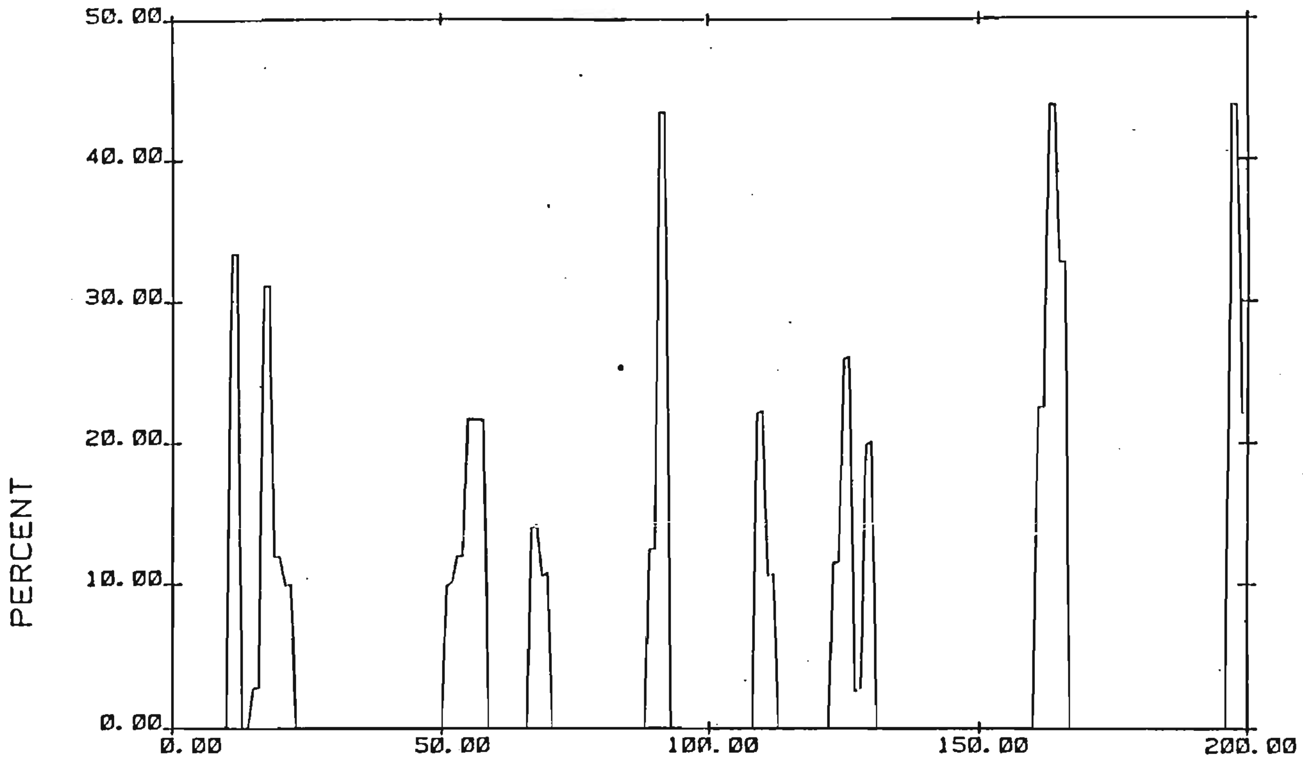
Utilization graph of ISP$_2$ - Test 2



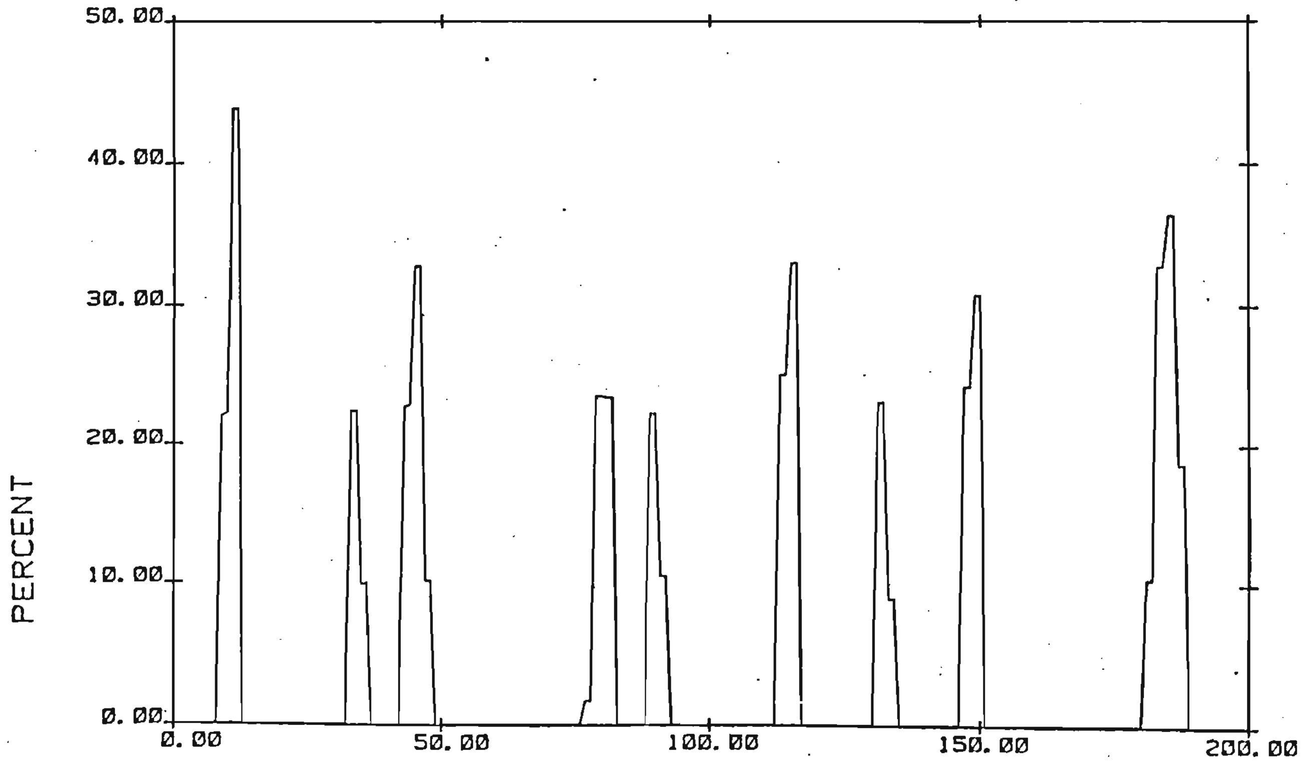
Utilization graph of ISP$_2$ - Test 3



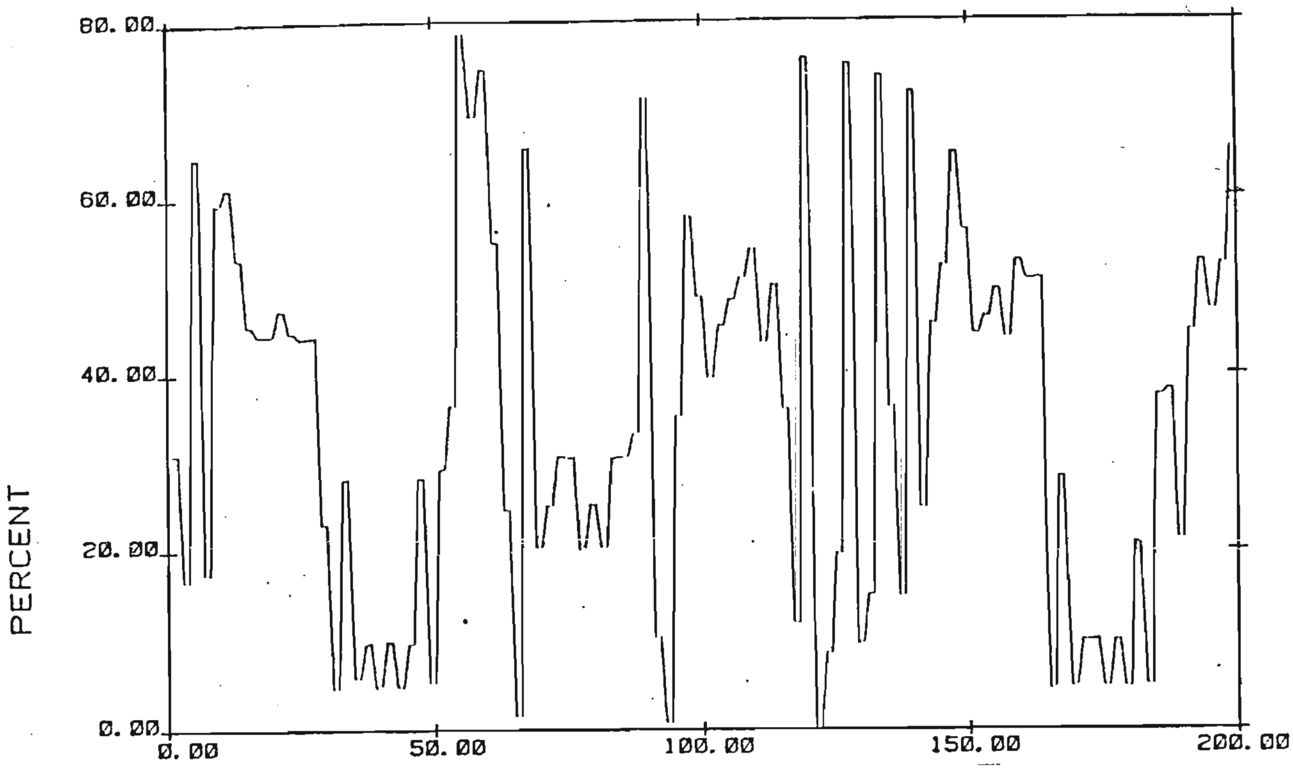
Utilization graph of ISP$_2$ - Test 4



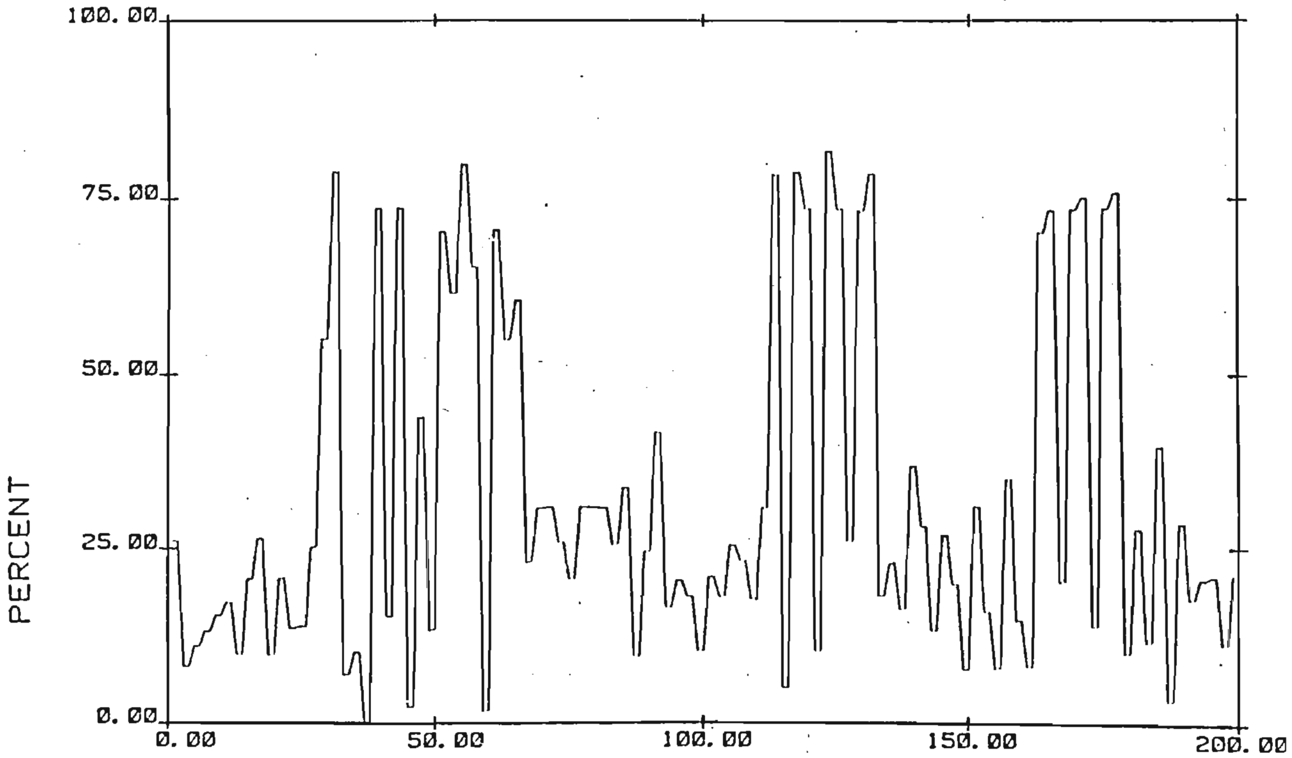
Utilization graph of ISP_2 - Test 5



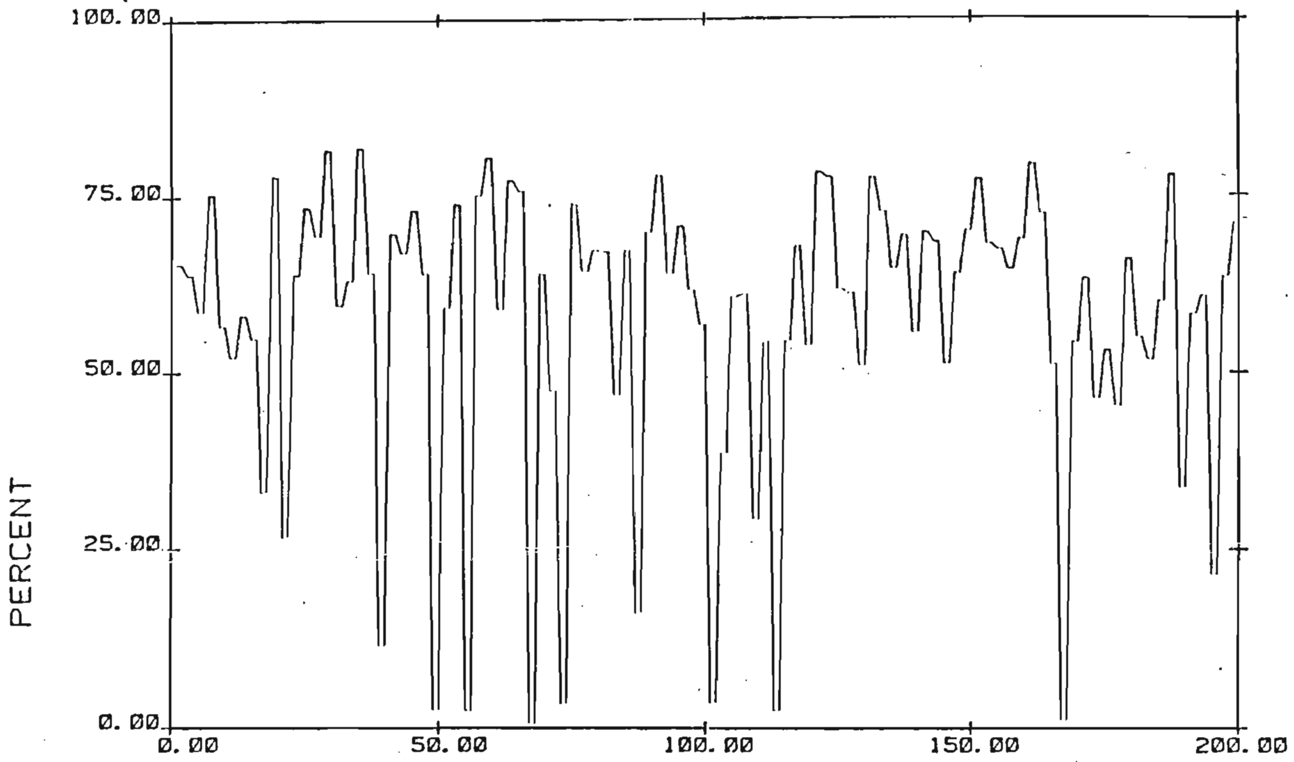
Utilization graph of ISP_2 - Test 6



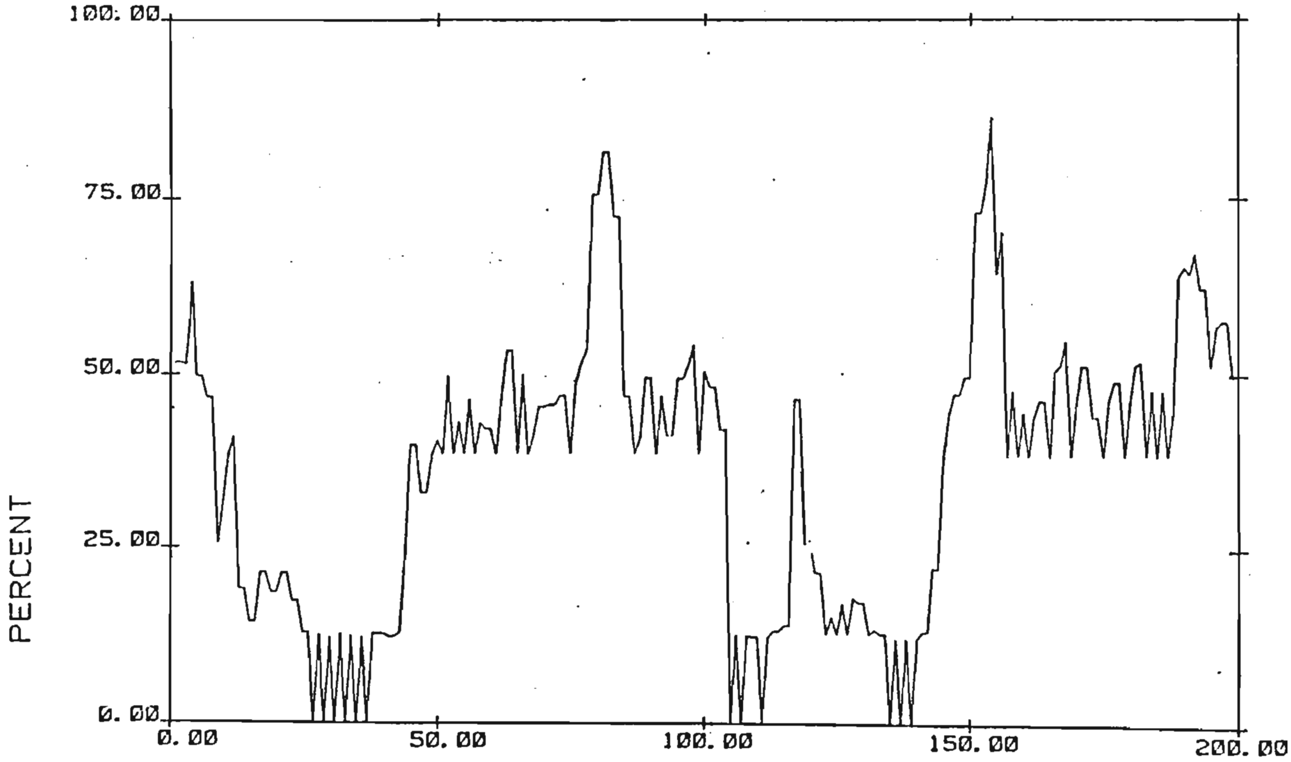
Utilization graph of ISP_2 - Test 7



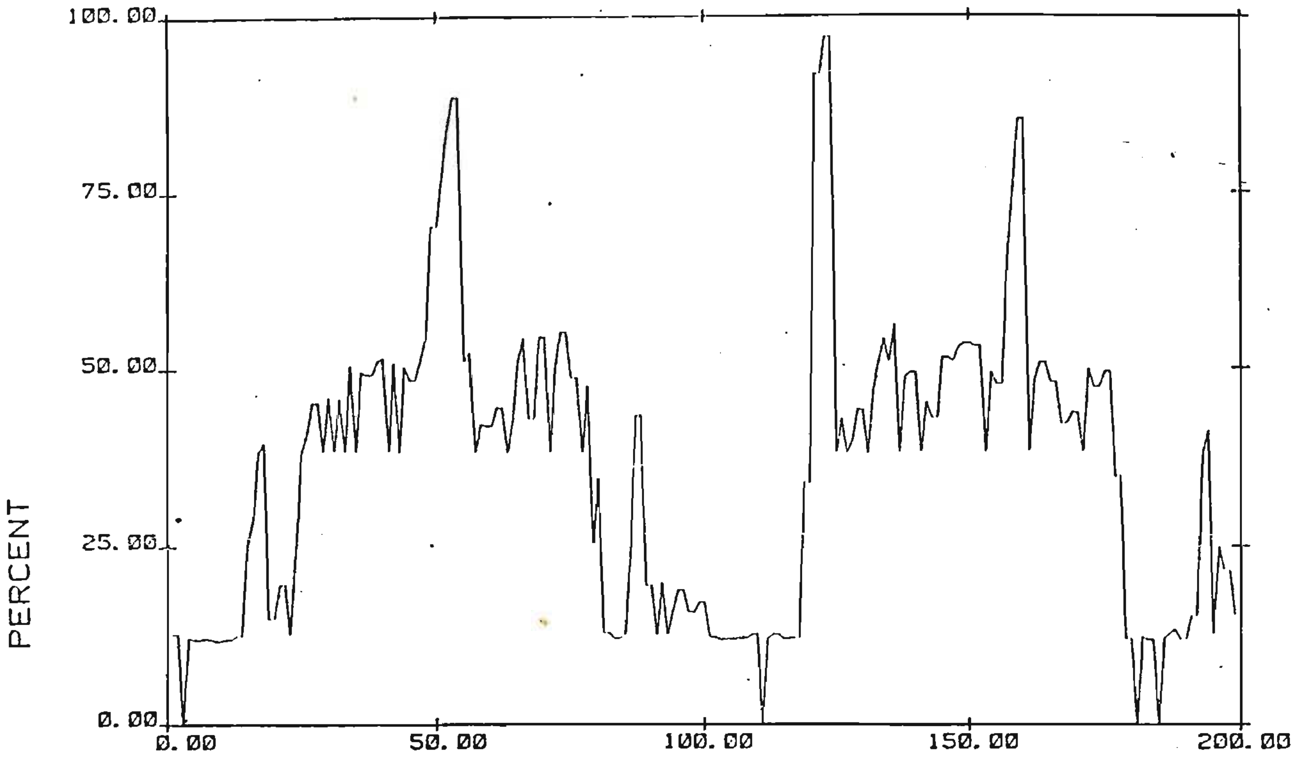
Utilization graph of ISP_2 - Test 8



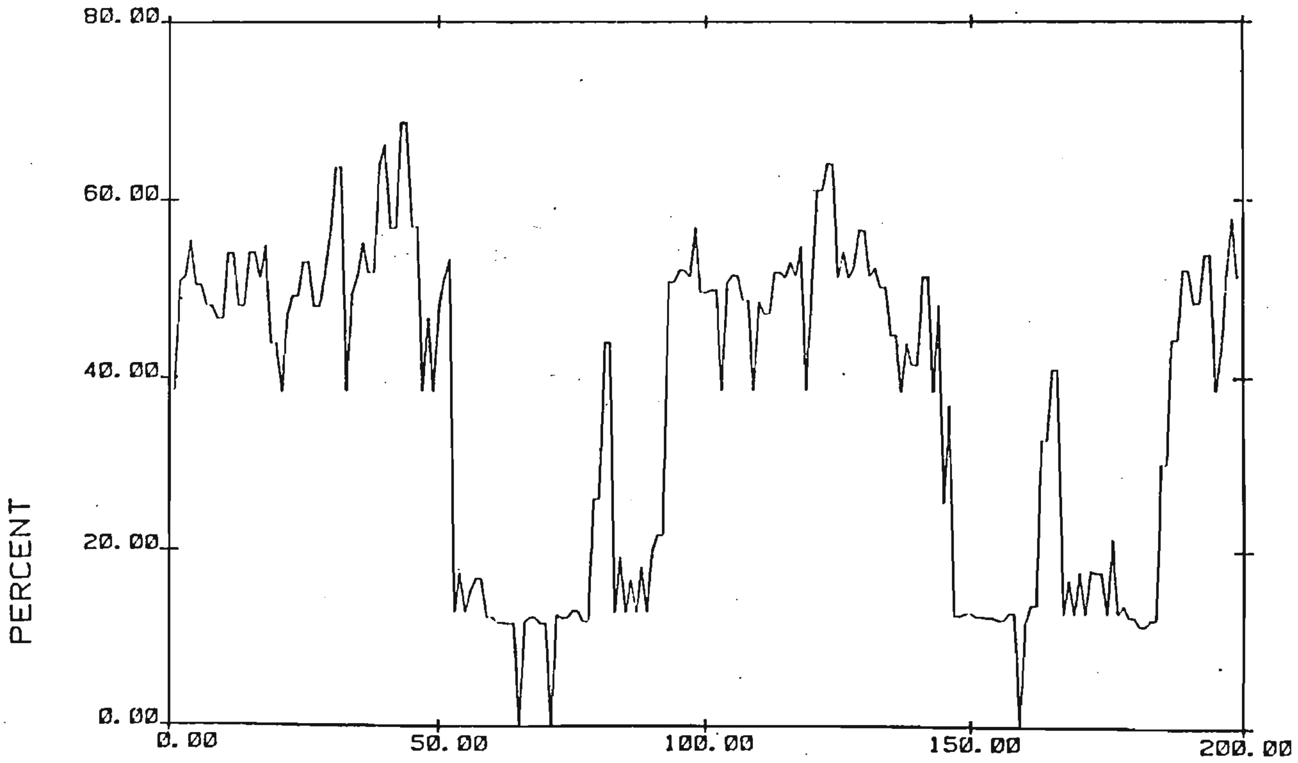
Utilization graph of ISP_2 - Test 9



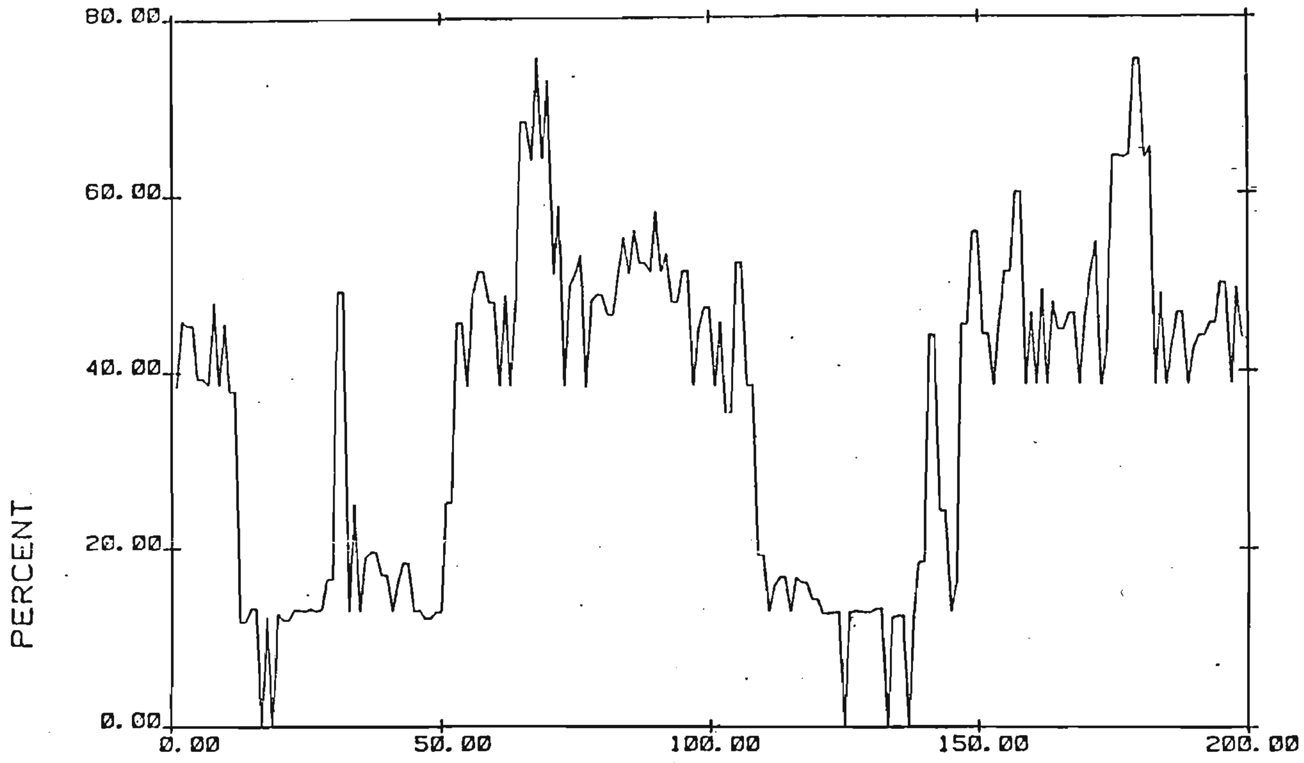
Utilization graph of IAP_1 - Test 1



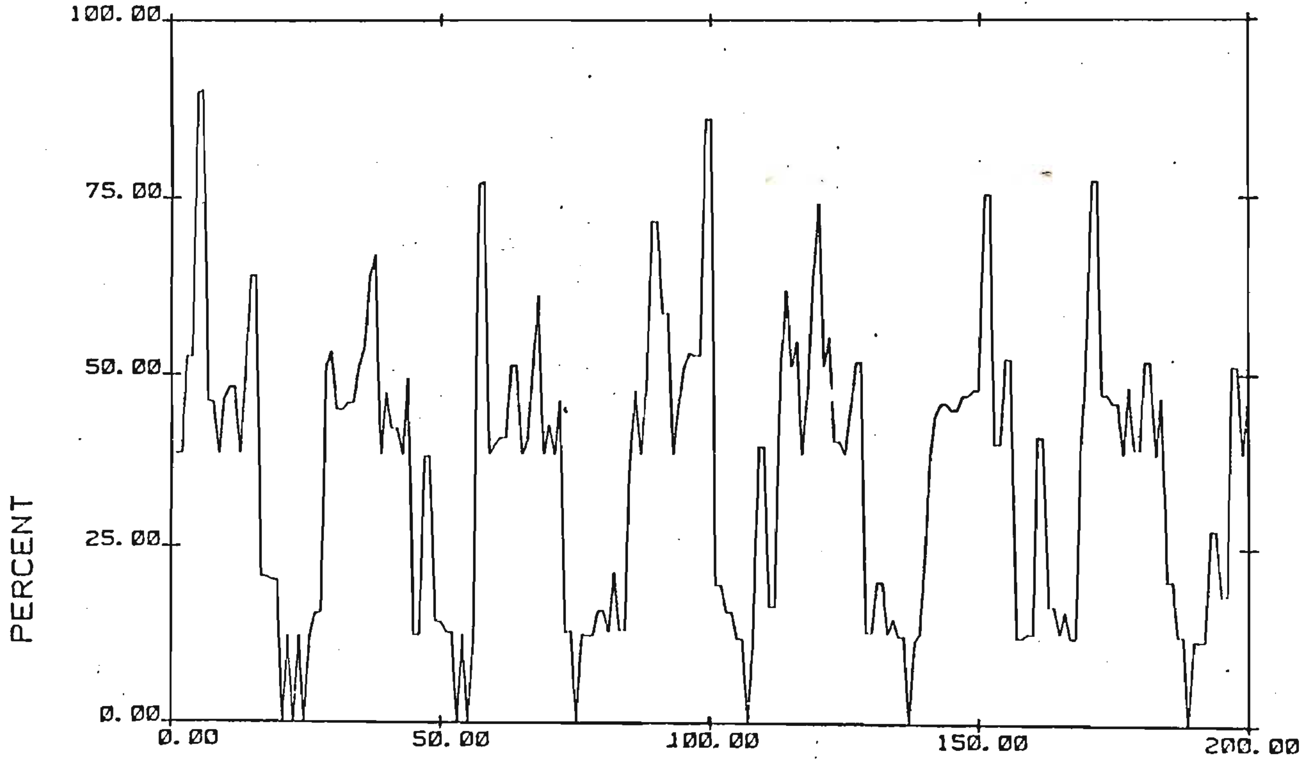
Utilization graph of IAP$_1$ - Test 2



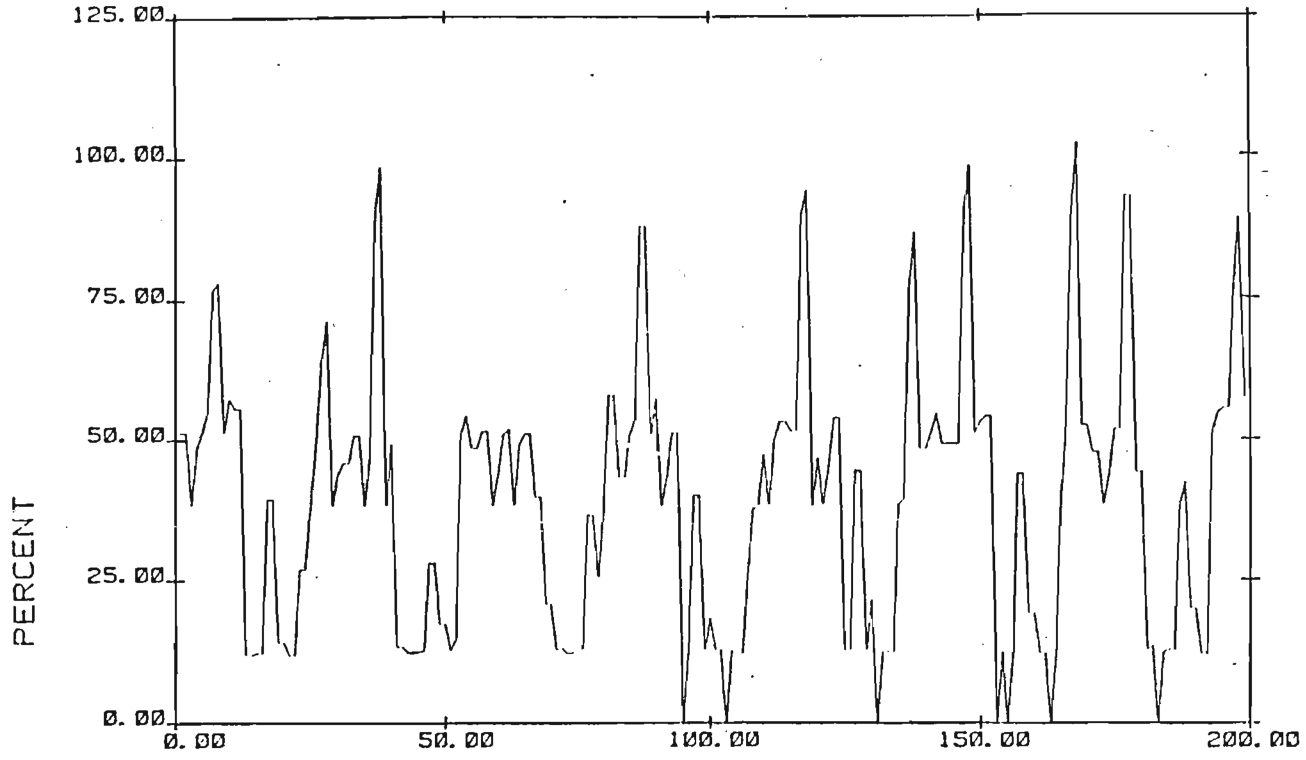
Utilization graph of IAP$_1$ - Test 3



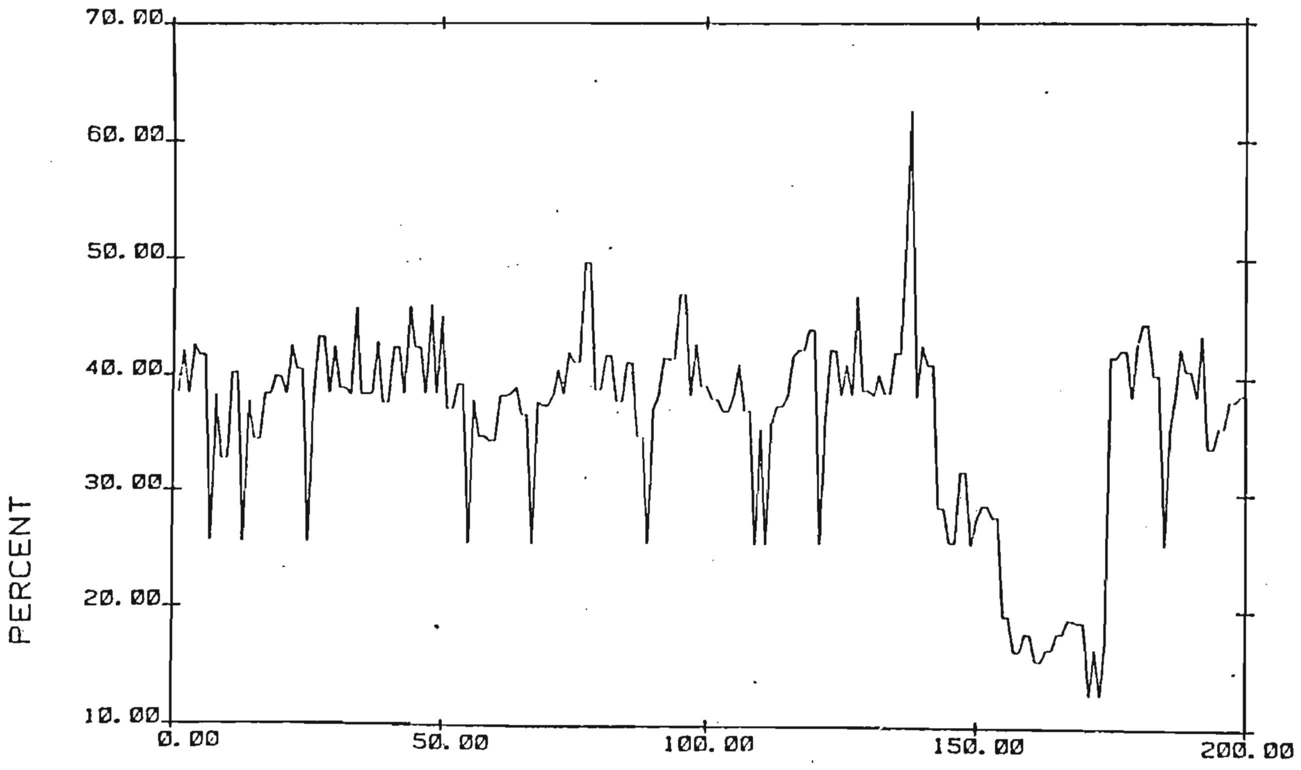
Utilization graph of IAP_1 - Test 4



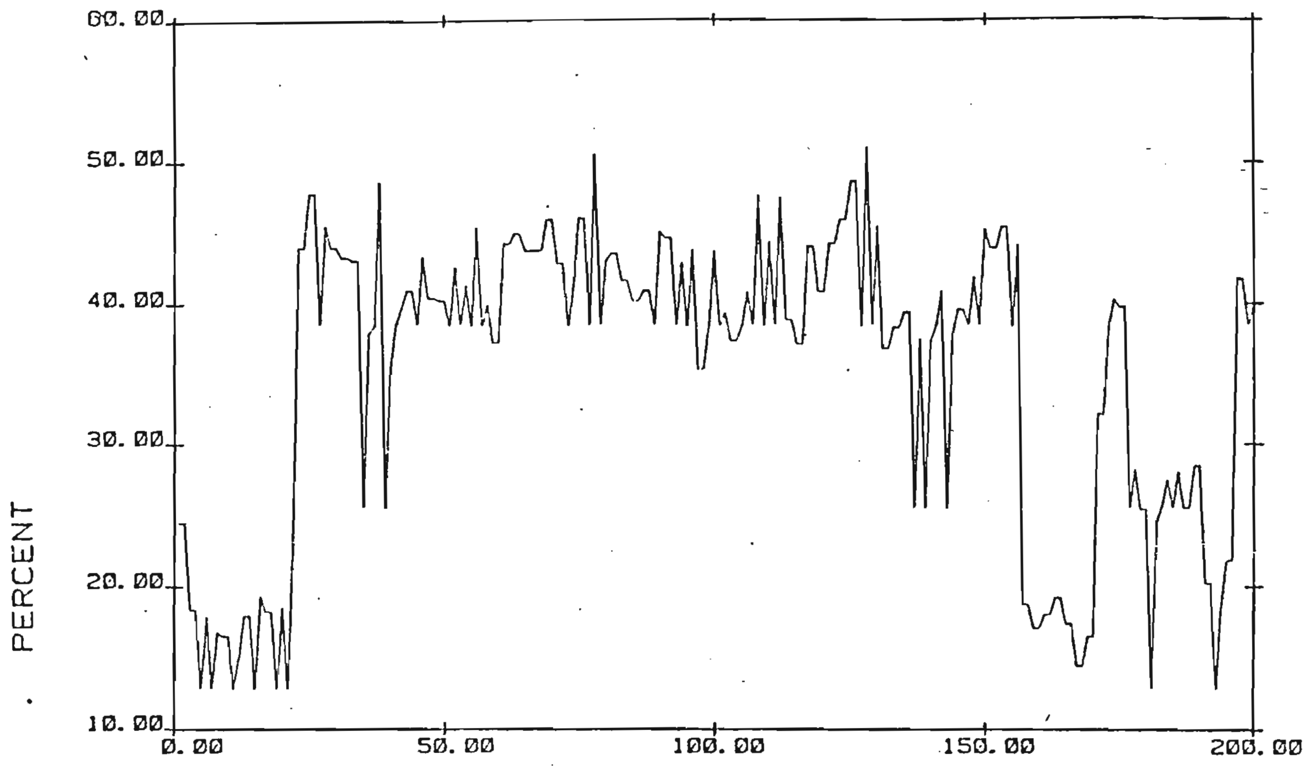
Utilization graph of IAP_1 - Test 5



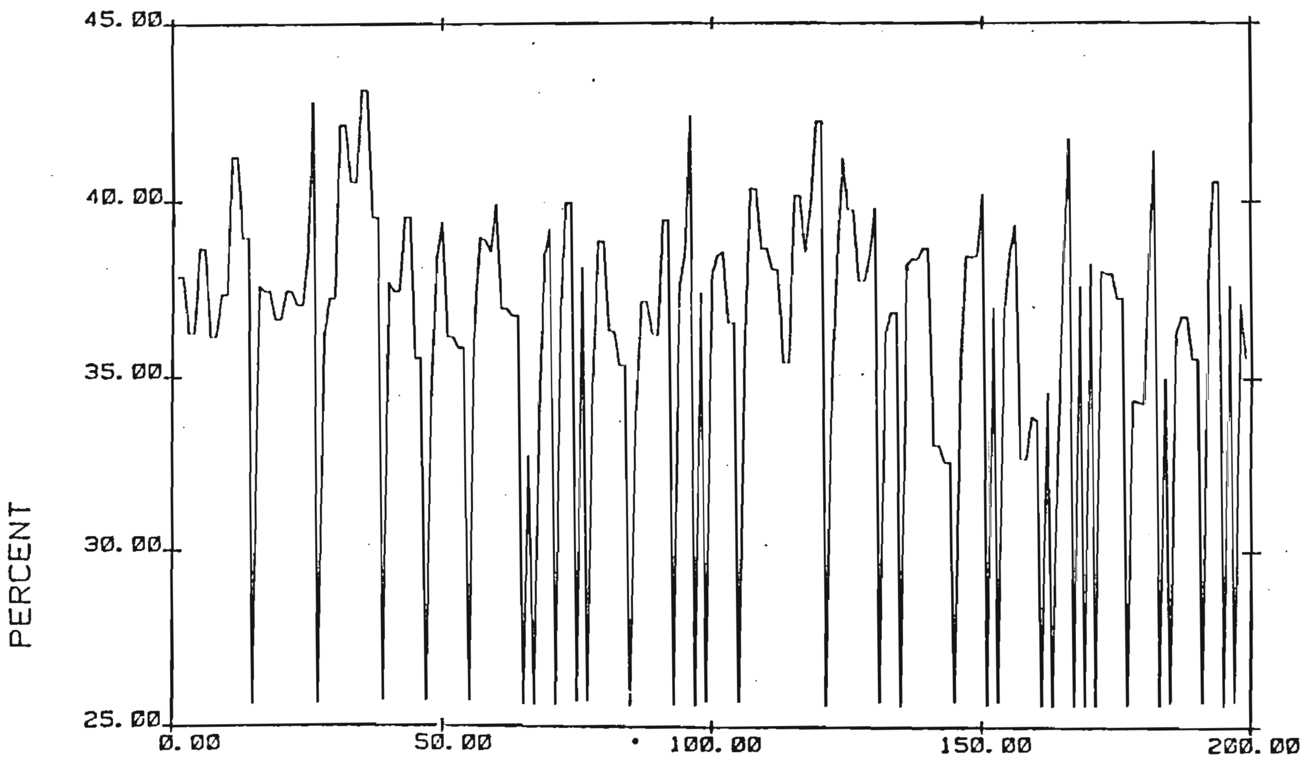
Utilization graph of IAP$_1$ - Test 6



Utilization graph of IAP$_1$ - Test 7



Utilization graph of IAP_1 - Test 8



Utilization graph of IAP_1- Test 9

APPENDIX D

EXPERIMENTAL METHODOLOGY

This appendix describes the methods used to obtain the results included in the thesis. A step by step description is given and the approach used would form the basis of any actual attempt to use the system described to carry out performance optimisation on a real computer system.

1. Determining the System Structure

This consists of analysing the computer system to be optimised and identifying the resources available within it which are available to the workload for utilisation. In the test system it consisted of:

(a) The Central Processor

(b) A fixed number of memory partitions constituting the internal memory

(c) A hard disk of 25Mb with DMA channel which was the external memory

(d) Three VDU terminals linked to buffered interface cards making up the Input/Output capability

(e) The operating system with file manager, scheduler, I/O drivers and utilities constituting the System Capability Particular elements.

This gave a system structure expressed as below

$$SYS = \begin{bmatrix} IPP_1 & 0 & 0 \\ ISP_1 & USP_2 & 0 \\ IAP_1 & IAP_2 & IAP_3 \\ SCP_1 & 0 & 0 \end{bmatrix}$$

2. Workload Characterisation

Characterisation of the workload consists of four steps. Firstly the elements or programs contained in it are identified. (In the case used here this is known but the program developed to record the workload was written in such a way as to be able to identify and sort the various programs recorded.) Next the frequency with which a particular program was the scheduled program was recorded. This was done using a program which interrogated the scheduler at fixed intervals of 10, 20, 30, 40 or 50mS. The data was then recorded on a magnetic tape. After these recordings had been made over a fixed period which could be equated to a typical measurement run, the data was reduced to a usable form. This consisted of reading the magnetic tape, sorting the recordings by program type, totalling the number of readings per program and normalising the value by dividing by the total number of readings.

This then resulted in the program significance factors. The method used was tested to ascertain the relative accuracy by using various sampling frequencies as well as recording periods and the accuracy was confirmed to be within 1% for repetitive recordings of the same workload.

Program histograms were now recorded for each program using the COMPAS system. Once again a number of recordings were taken and the correlation coefficients for successive runs determined. The coefficients were in all cases found to exceed 0,99 implying a relative measurement accuracy error of less than 1%. Using these histograms and the memory maps and program listings each point in memory was linked to a particular action or program routine which was in turn linked to a resource utilisation. The actual relative importance of resources was then determined by integrating the area under the histogram for each resource.

The memory importance for each program was not handled in this way. Instead the value was simply determined by dividing the actual memory size requirement by the size of the smallest partition. The five need factors were then normalized giving the results shown in Table 8 in the text.

It should be noted that the histograms taken included operating system activity directly generated by the program in question and this activity was assigned to SCP_1 for each program.

In the test case the program priorities were all equal so the priority factor P was made 1. In a real case the relative priorities for each program would have to be normalised and the resulting value used instead.

The need factor, significance factor and priority factors were now multiplied out to give the workload constants for each system particular element as shown in the equation below:

$$\begin{array}{ccccc} & & \begin{array}{ccc}\text{CRNCH} & \text{FILE} & \text{RESP}\end{array} & & \text{TOTAL} \\ \begin{bmatrix} KP_1 \\ KS_1 \\ KS_2 \\ KI_1 \\ KI_1 \end{bmatrix} & = & \begin{bmatrix} 0{,}228 & 0{,}0958 & 0{,}0923 \\ 0{,}0801 & 0{,}0319 & 0{,}0130 \\ 0{,}1373 & 0{,}0426 & 0{,}0 \\ 0{,}0114 & 0{,}0665 & 0{,}0373 \\ 0{,}1144 & 0{,}0293 & 0{,}0194 \end{bmatrix} & = & \begin{bmatrix} 0{,}4169 \\ 0{,}1250 \\ 0{,}1798 \\ 0{,}1152 \\ 0{,}1631 \end{bmatrix} \end{array}$$

for the case with maximum system memory configuration.

3. Utilisation Measurements

The structure of the system and workload descriptions are now complete. All that remains is to measure the utilisation of the various resources. The ideal would be to measure the workload and all the utilisations at the same time. This was not, however, possible with the equipment used in this study.

Firstly the workload measuring technique was not invisible to the system and secondly the COMPAS system did not contain multiple comparators, counters and timers in the form in which it was available for this study. For this reason a constant workload was chosen and all the measurements were carried out sequentially.

Firstly each type of measurement was tested for relative accuracy by measuring a specific action repeatedly and comparing the results. From this it was determined that the measurement of processor idle time, DMA channel throughout in unit time and I/O channel throughout was accurate to better than 1%. Memory utilization accuracy was found to be accurate to approximately 2%. The various tests were now carried out.

Firstly the system was configured as required for the test. The six programs making up the dummy workload were then loaded and time scheduled to run at fixed constant intervals. The COMPAS system was then set up and time scheduled to record data for a fixed period after the workload had been running for a period of time to ensure that the system had settled.

In the case of processor throughput the system wait loop was monitored and the time that this loop was active in unit time was repeatedly recorded. Each set of data was recorded over a 15 minute period. This data was stored on magnetic tape and later reduced by calculating the mean and the standard deviation over all the recordings, approximately 100 000 readings per final value.

Next the COMPAS was set up to record the time that the DMA channel was active in unit time in order to obtain the values required to ascertain the utilization of external memory. These data records were reduced in the same way as those for processor idle time. The time of activity was measured as the time from the I/O interrupt initialising the DMA transfer to the interrupt terminating the transfer.

Lastly the COMPAS was set up to measure the I/O channel utilisation. Here again the timing was based on the interrupts but the intérrupt daisy chain was monitored to determine when the interrupt began and when it ended. This was valid as the RTE IVB operating system works in such a way that the daisy chain is broken for the duration of an I/O transfer. The recorded data was once again reduced to a mean value and standard deviation.

Finally the memory utilisation was determined by using a small high priority interrupt driven program which integrated the system tables at a fixed time interval (20ms in the test case) to determine how many of the available partitions in memory were occupied. Again the recorded data was written to magnetic tapes as in the other cases and reduced in the same way.

This then describes the procedure followed to obtain the results set out in the table and graphs included in this thesis and from which the conclusions were made. The same basic procedure would be followed in evaluating an operational system although it is to be hoped that the workload and memory recordings would be made using methods invisible to the computer system being investigated. Furthermore the measurement system should ideally have sufficient comparators, counters and times to make all the measurements at the same time.