PERFORMANCE EVALUATION OF MULTIPROGRAMMED SYSTEMS.

Samir N. Adhikari

A Dissertation

in

The faculty

of

Engineering

Presented in partial fulfillment of the requirements
for the degree of Master of Engineering at
Concordia University
Montreal, Canada.

April, 1975.

© Samir N. Adhikari 1975

PERFORMANCE EVALUATION OF MULTIPROGRAMMING SYSTEM.

Name: Samir N. Adhikari.

ABSTRACT

The performance evaluation problem is explained and defined. This endeavour has exposed the need for an indepth knowledge of a working multiprogrammed system. Hence, functional aspects and design considerations of the operating system, Kronos, are discussed. Modelling is a very important and convenient approach to resolution of performance evaluation problem. Two different approaches to modelling exist; one is analytical model and the other is a logical model. The reviews of different analytical models are presented. Logical models are investigated crucially and the logical model for a simple operating system is built and simulated.

The results of the simulation are discussed to isolate the critical and the non critical parameters affecting the performance of multiprogrammed systems.

Most of the operating systems, marketed by different manufacturers, differ to varying degrees, Hence, importance is focussed on approaches to model building, rather than building a very general model encompassing all operating systems.

ACKNOWLEDGMENTS.

The author wishes to extend sincere thanks to:

- Prof. G. Martin of Sir George Williams University, author's advisor, for his guidence and counseling in this work.
- The department of Electrical Engineering for allowing this work to be carried out using department's resource and facilities.
- Computer centre staff for their numerous valuable suggestions.

CONTENTS.

CHAPTER 5-- PERFORMANCE EVALUATION THROUGH SYSTEM SIMULATION. DEVELOPMENT, BUILDING AND TESTING OF MULTIPROGRAMMING SYSTEM MODEL.

ILLUSTRATIONS.

CHAPTER 1

PERFORMANCE EVALUATION PROBLEM

1.1. Introduction to performance evaluation problem.

Modern times have seen an evolution of large systems using various sophisticated techniques, such as, multiprogramming, multiprocessing, segmentation and paging. But not all such systems are operating satisfactorily(D2). More and more people are taking a second look at these systems. Questions like the following are often asked:

(1) Is the system operating in optimum mode?

(2) Is the memory capacity adequate?

(3) How does the system performance change if the resources are altered?

(4) How does system behave if priority structure is altered?

and host of other questions.

Computer systems like other phenomenon in our times are growing bigger and bigger. The manufacturers of computers and systems are combining both batch oriented multiprogrammed systems with time-sharing systems. There are some interesting systems of this nature in market now, Kronos, Os/vs1, Os/vs2 and many other systems are functioning . As these systems cost millions of dollars; an urgent need is to evaluate these systems, and form a general theory of operating systems.

The performance evaluation problem is the problem of evaluating the efficiency of a large system. A large system's efficiency encompasses a large area of computer system design. It involves system hardware, system software, system job stream and optimum setting of number of parameters to achieve maximum efficiency. It involves complex interactions of various parameters, some of which can be controlled to any degree, some can be controlled to a limit and some can not be controlled. With such variablity in dimension of the problem, it is needless to point out the complexities involved. However, there is a pressing need of a systematic approach towards the solution.

An approach to find the solution of the efficiency problem needs a broader understanding of the system as well as the problem. In a broader sense, the efficiency of the system may be measured in terms of throughput (ie., jobs processed per hour). Associated with the efficiency problem is the problem of optimisation of throughput. This is a difficult problem because of the number of parameters involved, their complex interactions, and is further complicated by the unpredictability of the job stream.

The ideal job mix for a particular scheduling philosophy may not always be present and making scheduling so dynamic to change with every job is not efficient; because, the scheduler, itself, is a overhead on the system.

It is difficult to define a scheduling strategy that optimises each individual shared devices and over all shared devices. Thus, operating system designers have tried to compromise for maximum efficiency.

Two critical shared resources of the system are I/O devices and real storage. Dr. Denning(D2) has indicated how the theory of demand and supply could be applied to tackle the problem of critical resources.

The approaches to solution systematically resolve the critical and non critical factors involved in the problem. Out of various attempts to find the solution, we may mention the following:-

(a) Hardware monitors stationed at strategic points in the system are gathering data for determination and optimisation of throughput (P1).

(b) Software monitors are plugged-in with operating systems for supplying data for system performance evaluation and control.

(c) Analytical modelling of the system.

(d) Logical modelling of the system along with simulation.

Principally the present work involves with logical modelling of an operating system and simulation; it includes also to some extent analytical modelling.

Analytical models of the multiprogrammed systems suffer from oversimplifications. This is due to structural complexities of the problem. Although, the analytical models make assumptions about the system to reduce complexities; they are essential in system logical model.

1.2. Plan of the thesis.

The dissertation comprises of six chapters. Chapter 1 discusses multiprogrammed system's evaluation problem in general. Chapter 2 describes the multiprogrammed and time-shared system, Kronos, in brief. Chapter 3 delves into analytical model of the system. The reviews of many articles on this field by various contributors are included. Chapter 4 describes parameters that influence a multiprogrammed system. Chapter 5 is dealing with multiprogram system modelling and simulation. A system model is developed ; the system is simulated. The results showing the interactions of different parameters with objective functions are provided. Chapter 6 contains the conclusion.

CHAPTER 2

MULTIPROGRAMMED SYSTEM DESCRIPTION.

2.1. Definition of multiprogramming.

Multiprogramming is the art of resource sharing among various programs to achieve an effective resource utilisation in a computer system. The basic structure of all multiprogrammed systems could be divided into the following divisions:

(1) System monitor.

(2) Job scheduler.

(3) Allocation of main memory.

(4) Allocation of Cpu time slice.

(5) Allocation of peripherals.

(6) Data collection for statistical purpose.

A typical multiprogrammed-time shared operating system, Kronos at SGW, has been chosen for research in this project. Hence, instead of describing the system in general, the multiprogrammed system description is provided in context of Kronos.

2.2. SGWU system CDC 6200.

The configration of the system is shown in figure 4.1. It consists of one Cpu and ten peripheral processors, all of which share a large 60 bit word central memory. The Cpu minor cycle is 100ns. and the Ppu minor cycle is 1 microsecond.

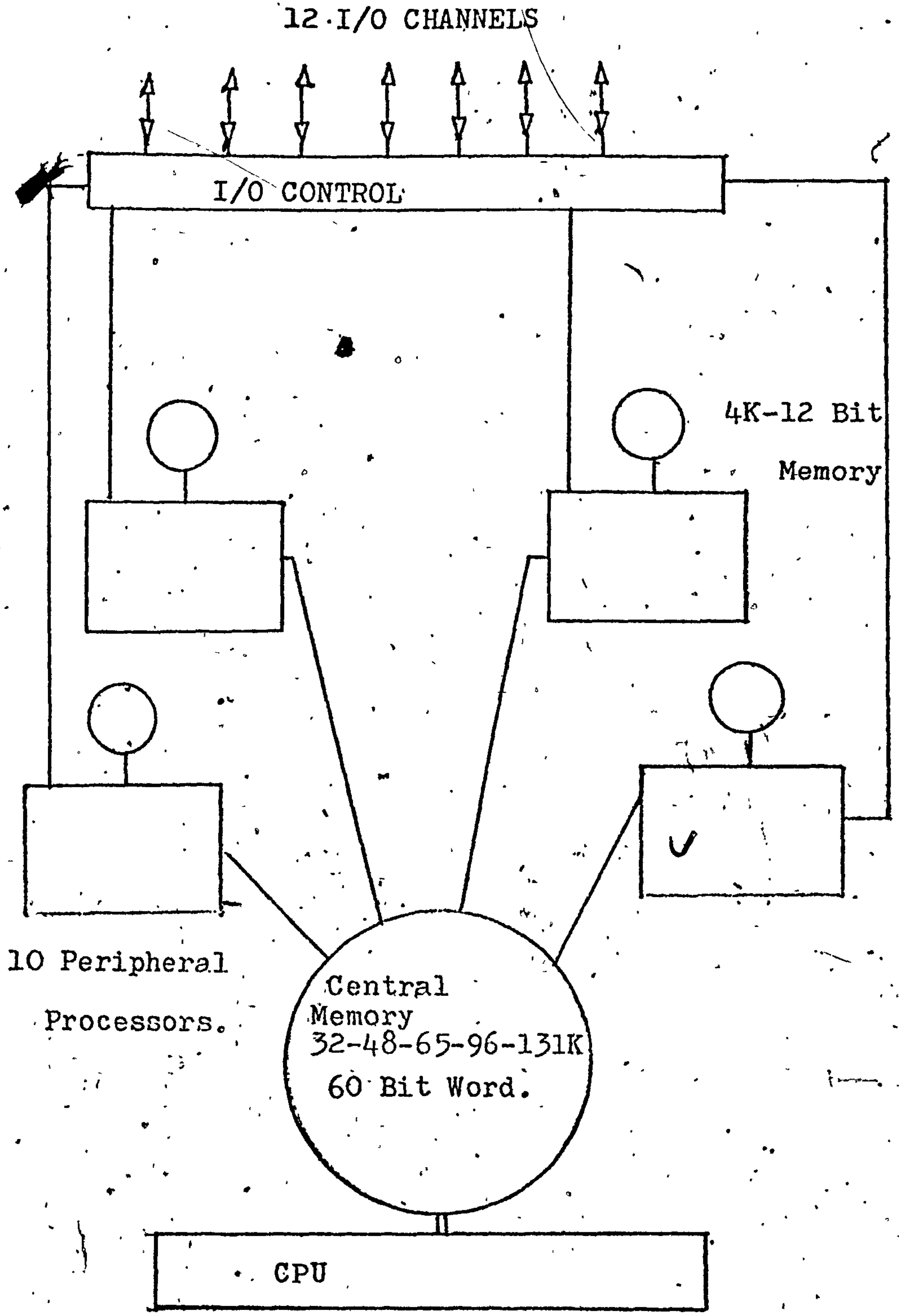


FIG. 2.1 Outline of System 6000

Each peripheral processor has 12 bit word 4K memory. There are 12 bit wide 12 data channels which are shared by PPUs. The main function of PPU is to operate I/O devices. They communicate with CPU through central memory words. The central memory in SGWU is 98K, 60 bit words with access time of 100ns., and it is organised in phased banks.

In this system the functions of CPU and PPUs are distinctly different. All I/O activities are performed by the peripheral processors, while the processing is done by the CPU. The system monitor is residing in one PPU, and one PPU is driving the operator display console. The rest of the PPUs are free to be assigned by the monitor for different I/O tasks. Each PPU has a store of 4K and some basic software such as overlay control and Disk driver etc. A system diagram is provided in figure 2-1 from this view point. A PPU communicates with the CPU by means of Exchange jump instruction.

When a job is selected for running, it is assigned system resources and CPU through some control locations in central memory called 'control points'. These are central memory resident and 128 words long. They contain various job parameters, eg., CPU time slice, Central memory time slice, CPU priority, Job priority, Initial and Final priorities for job type and the job exchange parameter area.

Kronos can support 26 control points. One control point is reserved for system monitor, one is for driving DSD display, one for remote batch terminal control and the rest are used as a system pool. The control points are assigned to jobs as required from the system pool. The control points used to be vital in CDC master and Scope operating systems(A1). But the problems have been cleared in Kronos.

2.3. Central memory organisation and control points.

Central storage is divided into user portion and resident system area. The resident system contains allocation tables, routine and file directories, a small amount of system CPU code, job control blocks and a set of peripheral processor routines used for overlays. Job control blocks or control points occupy 128 words each. The control point is important to the system. All resources for running a job are assigned to the control point and the CPU is scheduled to the job by 'Exchange Jump' instruction. Exchange package area of the control point is interchanged with data from the interrupted program, and the size of the area is 16 words which contain the register values. The system has a very fast swapping time which is about 2 microsecs. Job control parameters are also stored in this area. The buffer size is about 72 words.

2.4. Job scheduler.

The job movement strategy in Kronos is controlled by the Job scheduler. Job scheduler is a program which is scheduled by the monitor according to a control parameter in the system. The parameter can be modified through operator's command.

The function of this program is to take a snapshot of all control points in central memory. The priorities of jobs that are presently running in the system are compared with the jobs that are waiting in the queue. If a job waiting in the queue has higher priority than a job running in the system, and if the system resources required by the waiting job could be met, a rollout sequence is initiated. The rollout, which is known as Autoroll in Kronos, has been described in detail in Rollout mechanism. It is mentioned that job scheduler program is initiated periodically but following conditions will trigger scheduler as well:

(a) A job finishes or aborted.

(b) A new job arrives in Queue.

2. 5Job Priority.

Each job in the system carries a single twelve bit priority called queue priority. A large value signifies high priority and a low value signifies low priority. Several priority classes and values are reserved for identifying jobs in special states, such as being rolled out or rolled in or waiting for some operator action. Each time job scheduler executes, it takes a snapshot of control points. This provides the list of jobs that are running, the resources thay are using and their priorities. The scheduler compares the waiting jobs with their resource requirements with the running jobs in a decreasing order of queue priority. In a case when sufficient resources are available, ie., not in use by other jobs, the scheduler assigns a peripheral processor to rollin the job.

When a job waiting in queue but having priority higher than the one in the system, requires resources that could be met if the occupying job frees them; the system will force a rollout on the occupying jobs.

A rollout file contains every important information of a job; central memory image, the control point status along with its 128 words buffer, file pointer space, but magnetic tape unit associated with the rollout job remains with it for practical reasons. There are some controls which are exercised over rollout densities. The number of concurrent rollout is limited to two jobs. When a rollout sequence is started, the scheduler continues to search the job queue until exhausted. It could start another rollout sequence if needed.

When the rollout sequence is ended the scheduler is called in. It ignores the job which started the rollout sequence, and proceed normally comparing running jobs with waiting jobs. This makes the system very dynamic in response to system changes. For instance if a job enters the queue that has higher priority than the one which started the rollout sequence , the higher priority job is executed first, or supposing a job finishes before the rollout is complete then the waiting job would be scheduled immediately to save unnecessary delay.

2.6: Queue priority manipulation.

A system overlay, that executes at time which is a submultiple of scheduler, adjust the queue priorities of all jobs. Each job depending on its origin and queue type has entry queue priority. Some typical values are given in Table 2.1. The values shown in Table 2.1 could all be modified by operator's command.

Kronos avoids locking of low priority jobs by high priority jobs through priority aging mechanism. The priority of an waiting job is modified by priority increment program. Each job origin code has eight words assigned to it in central memory resident (CMR). The parameters are shown in Table 2.2. The Table shows that there are upper bound and lower bound for every type of job.

TABLE 2.2. Job origin and Q-priorities.

JOB ORIGIN	Q-TYPE	PRIORITY
SYSTEM	INPUT	6600
	ROLLOUT	6000
	OUTPUT	400
BATCH	INPUT	2400
	ROLLOUT	2400
	OUTPUT	200
EXPORT/ IMPORT	INPUT	3400
	ROLLOUT	3400
	OUTPUT	200
TELEX	INPUT	4000
	ROLLOUT	4004
	OUTPUT	200
MULTI-TERMINAL	INPUT	6774
	ROLLOUT	6774
	OUTPUT	6000

TABLE 2.3. Queue priority of different job origin.

INPUT QUEUE	OP	LP	UP	IN	COUNTER
ROLLOUT	OP	LP	UP	IN	COUNTER
OUTPUT	OP	LP	UP	IN	COUNTER
CONTROL POINT	Pri	Cputs		Cmts	
MEMORY	Mnj	Mfts		Mflt	

OP: Entry Priority.

LP: Lower Priority.

UP: Upper Priority.

IN: Increment.

Pri: Initial Cpu Priority.

Cputs: Cpu time slice.

Cmts: Central memory time slice.

Mnj; Maximum number of jobs.

Mfts: Maximum field length of any job of this origin.

Mftl: Maximum total field length for all jobs of
this origin.

TABLE 2.4. Time-slice and CPU priority for different job origin.

JOB ORIGIN TYPE	TIME-SLICE CPU(MS.)	INITIAL CPU PRIORITY.
SYSTEM	100	1
BATCH	400	30
EXPORT/IMPORT	400	30
TELEX	40	30
MULTITERMINAL	400	31

If the priority of the job is between these limits, it is aged upwards. Whenever the priority-increment-program is activated, the value in the control byte of the job is incremented. When the incremented value is equal to some set value, the priority of the job is incremented. This helps the job moving through the system, and avoids the jamming of lowpriority jobs in the system.

2.7 CPU Scheduling.

The CPU is scheduled in round robin fashion, and is given to the job according to priority assigned to the job at the control point. The CPU priority and time-slice in ms., given to jobs of different origin, are shown in Table 2.3. The distribution of time-slice among various jobs reflect the following objectives:

(a) High priority jobs should not monopolize the system.

(b) To let most batch jobs run to completion in one time-slice.

(c) For even distribution of compute time to different terminal users; time-slice for Telex jobs are set low.

(d) Batch jobs have large time-slice.

When a job reaches the end of its time-slice, its priority is reduced to lower bound so that jobs waiting in rollout and input queues could force a rollout on the resource bound job. Thus the ratios of input queue entry priority to lower bound queue priority become significant for motion of jobs through the system.

The lower bound queue priority is lower than queue entry priority so that resorce bound jobs could be rolled out to make room for waiting jobs. The rolled jobs ages in rollout queue . It is scheduled again when its queue priority is higher than all waiting jobs and its system resource requirement matches with what is available in the system at the moment of scheduling decision.

In Kronos Multiterminal jobs are given the highest priorities, because it is desirable to complete these jobs quickly. The best scheduling strategy is to give the cpu to the job which releases it fastest, and multiterminal jobs, in general, require very little cpu time.

The system jobs, running in the background, are given the lowest priority.

2.8. Rollout Mechanism.

The 4th word in job control area directs the control of a job while it resides at a control point in central memory. This word contains the following values:

- -byte 0: initial cpu priority set at job initiation or user log in.
- -byte 1: cpu time-slice ms./64.
- -byte 2: central memory time-slice, sec.
- -byte 3&4 not used.

There are three reasons for a job leaving a control point.

a) a job completes or aborts.

b) Terminal I/O is required.

c) The control point is made available for higher priority jobs.

We will discuss the type(c) here in detail. Everytime a job consumes its time slice, or central memory time slice, and its queue priority is in the range $100 < qp < 7770$, the queue priority is set to lower bound priority for input or rollout files for that origin types. This value is lower than the entry priority for input and rollout jobs. Thus any job in the queue with priority higher than the entry priority forces the resource-bound job to be rolledout. The rolled out jobs age until their priorities are higher than entry priorities and are scheduled again to control points.

Once a job is initiated to a control point, it is desirable to use the resources allocated before another job forces it out. Thus its priority is changed to upper bound for the origin and queue type. However, if the entry priority at the time of scheduling is greater than the upper bound priority, the job retains that value.

CHAPTER 3

MATHEMATICAL MODELS.

3.1. Mathematical models of multiprogramming systems.

Analytical models of the multiprogramming systems have been grouped into two categories, one using exponential distribution for job arrivals and queuing theory and the other using Markov chain models. In both cases the models have used assumptions for mathematical simplicity because it is extremely difficult to express the structure of the multiprogramming problem completely in mathematical terms. A logical model is well suited for this purpose.

3.2. Probablistic model of the system.

Fig. 3.1 shows the system representation for mathematical formulation. Different job types arrive at the system with different arrival rate $\lambda_1, \lambda_2, \lambda_3 \ldots \lambda_n$.

These jobs are first queued on disk. From disk according to priorities $\gamma_1, \gamma_2, \ldots \gamma_n$ jobs are brought into core for service. Only one cpu is available for service. In order to find the distribution of job arrival time, we would assume that the probability of a job arrival at a small interval is small but average rate of arrival is significant so that we can approximate arrival pattern by poisson distribution.

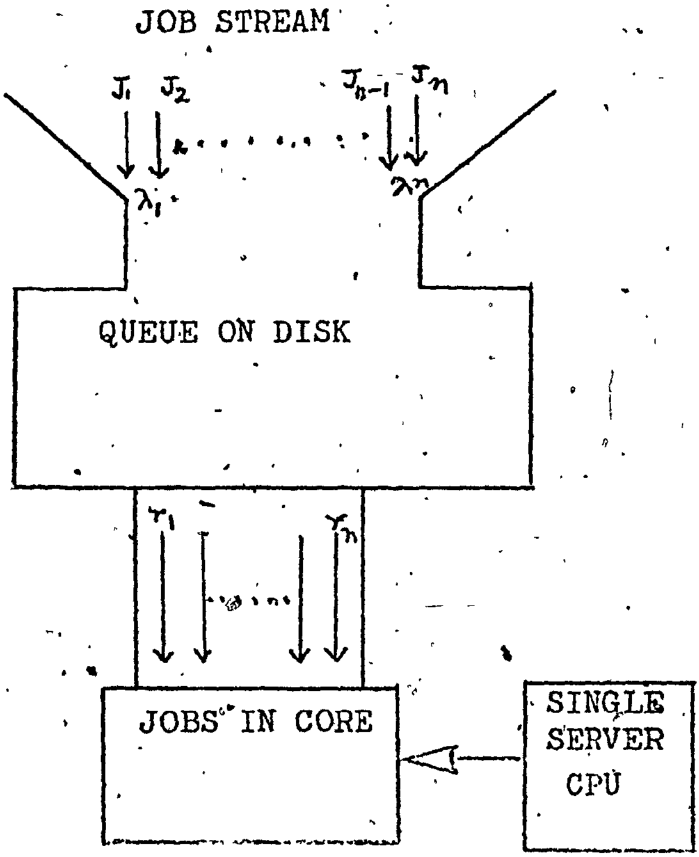


Figure 3.1. System representation for Mathematical formulation.

If X be the time interval between job arrival then poisson process provides the following probabilities:

$$P\left[X \leqslant t\right] = 1- e^{-\lambda t}; \quad \lambda = \text{Mean arrival rate.}$$

Let the probability of n number of jobs of priority γ_i in a given interval t is $p_{ni}(t)$ for n= 0,1,2.....∞

Thus $\quad p_{ni}(t) \geqslant 0$

$$\sum_{n=0}^{\infty} p_{ni}(t) = 1$$

$$p_{0i}(0) = 1 \; ; \; p_{ni}(0) = 0 \quad ; \; n > 0.$$

Let us have two non overlapping intervals t and τ and let k number of jobs of priority γ_i arrive in first and n-k number of jobs in second interval.

The probability of both happening is

$$p_{ki}(t).p_{n_i-k_i}(\tau)$$

Thus overall probability is

$$p_{n_i}(t+\tau) = \sum_{k_i=0}^{n} p_{k_i}(t).p_{n_i-k_i}(\tau) \text{ ------------(3-1)}$$

The general solution of the above equation according to (C6).

$$p_{n_i}(t) = e^{-\lambda_i t} \; \frac{(\lambda_i t)^{n_i}}{n_i!} \quad \text{------------(3-2)}$$

$$E[n_i] = \lambda_i t$$

The equation (3-2) is very important relation in queuing theory. In markovian modelling of queuing, the rate matrix is computed from equation (3-2) by differentiation.

Let the service time probability distribution be also exponential. In actual practice the service time distribution may not be exponential but we have simplified the case and assume it to be exponential.

Let the density function of service time distribution be $s_0(t) = e^{-\mu t}$ ------ (3-3)

$s(t) = \mu e^{-\mu t}$-----------(3-4)

Where $\mu = \frac{1}{T_s}$

T_s = Average service time.

The previous mathematical exercise is provided to form certain queuing discipline such as arrival of jobs and service time distribution to be exponential. This is the case in probablity theory when events in a sample space are due to numerous and widely varying factors, and independent.

The objective of the model is to provide an insight into how jobs of different priorities $r_1, r_2 \cdots r_n$ having arrival rate $\lambda_1, \lambda_2, \cdots \lambda_n$ and service time $\mu_1, \mu_2 \cdots \mu_n$ accumulate in disk queue. The scheduler could compare the design limit of waiting times and actual values and when exceeded could alter the job selection criterion and bring the system back into balance.

Let the priority number be 1, 2, 3,........,n with lower number corresponding to higher priority. Let $\lambda_1, \lambda_2 \cdots \lambda_n$ be the input traffic densities each according to a poisson process. Let $S_k(x)$ be the service time distribution of K^{th} priority.

Then the $1^{st.}$ and $2^{nd.}$ moment of the waiting time of the K^{th} priority(M2) are the following:

$$W_k^{(1)} = \frac{\sum_{i=1}^{n} \lambda_i \alpha_i^{(1)}}{2\left(1-\sum_{i=1}^{K} \lambda_i \alpha_i^{(1)}\right)\left(1-\sum_{i=1}^{K-1} \lambda_i \alpha_i^{(1)}\right)} \text{-----------}(3\text{-}5)$$

$$W_k^{(2)} = \frac{\sum_{i=1}^{n} \lambda_i \alpha_i^{(3)}}{3\left(1-\sum_{i=1}^{K} \lambda_i \alpha_i^{(1)}\right)\left(1-\sum_{i=1}^{K-1} \lambda_i \alpha_i^{(1)}\right)^2} + \frac{\sum_{i=1}^{n} \lambda_i \alpha_i^{(3)}}{2\left(1-\sum_{i=1}^{K} \lambda_i \alpha_i^{(1)}\right)^2\left(1-\sum_{i=1}^{K-1} \lambda_i \alpha_i^{(1)}\right)^2} + \frac{\sum_{i=1}^{n} \lambda_i \alpha_i^{(2)} \sum_{i=1}^{K-1} \lambda_i \alpha_i^{(2)}}{2\left(1-\sum_{i=1}^{K} \lambda_i \alpha_i^{(1)}\right)\left(1-\sum_{i=1}^{K-1} \lambda_i \alpha_i^{(1)}\right)} \text{----------}(3\text{-}6)$$

The average queue size :

$$L_k^{(1)} = \lambda_K W_k^{(1)} + \lambda_K \alpha_K^{(1)} \text{-------------}(3\text{-}7)$$

$$L_k^{(2)} = \lambda_K^2 W_k^{(2)} + 2\lambda_K^2 W_k^{(1)} \alpha_K^{(1)} + \lambda_K^2 \alpha_K^{(2)} \text{------}(3\text{-}8)$$

Where $\alpha_K^{(1)}$ and $\alpha_K^{(2)}$ are the $1^{st.}$ and 2^{nd} moments of service time distribution $S_k(t)$ of the K^{th} priority.

From (3-3)

$$S_k(t) = e^{-\mu_k t}$$

Density function $\frac{d}{dt} S_k(t) = \mu_k e^{-\mu_k t}$

$$\alpha_k^{(1)} = \int_0^\infty t\,\mu_k e^{-\mu_k t}\,dt$$

$$\alpha_k^{(2)} = \int_0^\infty t^2 \mu_k e^{-\mu_k t}\,dt$$

$$\alpha_k^{(3)} = \int_0^\infty t^3 \mu_k e^{-\mu_k t}\,dt$$

The equations (3-5), (3-6), (3-7), (3-8) are from reference (M2).

Thus we could find the average wait time of a K^{th} priority job and consequently the queue length of the K^{th} priority job in disk as design value. The above analysis is valid even if the service time distribution is not exponential.

3.3 Markov model of the multiprogrammed system.

Markov model for a multiprogramming system is very promising when we consider that the job arrival and service time demand on the system follow probablity distributions which are independent. Thus if we are interested in the net effect, such as number of jobs waiting in the queue or what is the average waiting time for a job, we may view the multiprogramming system as a stochastic process.

The characteristics of the process would be provided by a sequence of random variables X_1 , X_2 , X_3 , ...X_n with a continuous time parameter t. But we will make the continuous time parameter t to be discrete for the ease of analysis as t_1 , t_2 , t_3 , ...,t_n because we are interested in the change of the system.

The random variables X_1,X_2,X_3, ...,X_n could represent any desired event in the system's sample space. Thus to represent various events in the system we have to generate various functions of random variables. Are random variables X_1, X_2, ...,X_n independent? Obviously they are not because if they are there would have been no complexities in finding the joint probablity distribution of the sequence from the probablity product law, eg., p(A) = p_1 ; probablity of event A is p_1 , p(B) = p_2 ; probablity of event B is P_2 , then p(AB) ; probablity of event A and B both happening is p_1p_2.

Thus it would have been sufficient to merely know the probablity distribution of $X_1,X_2,X_3,\ldots\ldots X_n$ individually to predict the total system behavior. But what kind of dependencies exist between the random variables? In this analysis we would assume that dependency is markovian, ie., the probablity of next step only depends on the present position or the system is memoryless. Multiprogramming system fits this process very conveniently and appropriately. Jobs are arriving at a multiprogramed system terminal with interarrival time exponentially distributed. Jobs are read in the system and get their service and exit. Supposing we are observing the system over a period of time t and we are interested in the number of jobs in the queue. Let this is represented by the random variable X_t. Supposing we are only interested in X_t at epoch ie., when a job comes into queue or leaves the queue. Thus we have discreticized the continuous time parameter t into $t_1,t_2,\ldots.t_n$ and the random variable $X_1,X_2,\ldots.,X_n$. Now let us find the probablity that random variable $X_n=x_n$ when $X_{n-1}= x_{n-1}, X_{n-2}= x_{n-2}\ldots\ldots.,X_1=x_1$ or

$P\left[X_n=x_n \middle| X_{n-1}=x_{n-1}, \ldots.X_1=x_1\right]$ which is actually a conditional probablity. But the value of $X_n=x_n$ only depends on $X_{n-1}=x_{n-1}$ since the random variable is only representing the event of total number of jobs in the queue. Hence the probablity reduces to $P\left[X_n=x_n \middle| X_{n-1}=x_{n-1}\right]$

This is often represented as $p_{ij} = P\left[X_n = j \text{ when } X_{n-1} = i\right]$ and known as transition probablity and stochastic process is markovian.

In markov process once we know the transition probablities ie., probabliity matrix,

$$\pi_j = \begin{array}{cc} & j \rightarrow \\ i \downarrow & \begin{bmatrix} p_{11} & p_{12} & p_{13}\text{-----}p_{1n} \\ p_{21} & p_{22} & p_{23}\text{-----}p_{2n} \\ \text{-----} & \text{-----} & \text{-----} \\ \text{-----} & \text{-----} & \text{-----} \\ \text{-----} & \text{-----} & \text{-----} \\ p_{n1} & p_{n2} & p_{n3}\text{-----}p_{nn} \end{bmatrix} \end{array}$$

and the probablity vector $\bar{p}_0$, the behavior of the system could be predicted (B2).

$\bar{p}_0$= Initial probablity vector.

Supposing transition probablity P is given from state i. We are interested in knowing the state probablities after n transition. This is obtained by raising state probablities to the power of n, eg., P^n providing the transition probablities from state i → j in n transition. Unconditional probablity = $P_j^{(n)}(t)$ after n transition is provided by

$$P_j^n(t) = P_o P^n$$

or $\pi_j(t) = P_o P^n$

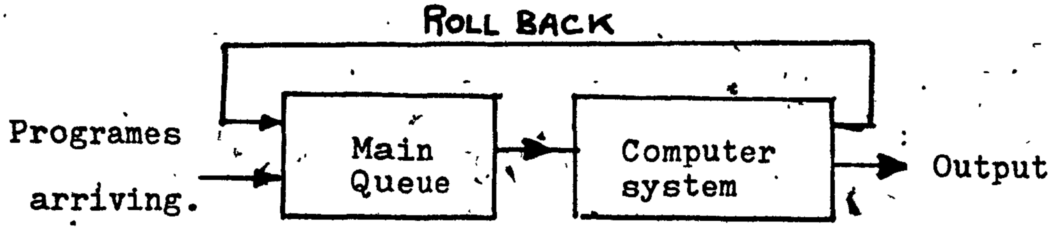
We also can find the limiting case or the steady state case $t \rightarrow \infty$ of the state of the system.

The object of the present analytical model to provide such average estimate of the system. Thus we are going to conclude our discussion of markovian process and proceed to the discussion of actual model.

3.4 System model.

Processing in all multiprogramming system is done by cpu in quantum time allotment ,ie., the system assigns a quantum q , to each processing program to balance service between a host of programes.

We will assume a very simple model for the system known as round robin queuing principle as proposed by Coffman(C4).



The programmes arrive in the system and queued in the main queue, the interarrival time is poisson distributed I(t) and is selected for service for quantum of q units of time, if service is not completed by that time it is rolled back to end of the queue and the process continues. We also assume that service time distribution is exponential S(t).

We would also assume that the average rate of service is faster than the average rate of job arrival which result in a statistical equilibrium of the system.

The object of the exercise is to find an estimate of the average number of jobs in main queue as a function of quantum time, program swapping time, etc. We also want to find the average waiting time of a program in the queue. The following parameters are defined below :

λ :average arrival rate of jobs.

μ :$\frac{1}{T_s}$ where T_s average service time.

τ : Program swapping time including executive overhead.

q : Quantum time slice.

t_k: epoch time , K= 1, 2, 3, ...

The random process we are interested is the number of jobs in the system as a function of time. We define this by $\xi(t)$ where t_k is the epoch time. We select t_k just after job expiration or quantum expiration. The distribution of time intervals (t_k-t_{k-1}) ,between successive epoch would be

$$F(x)=P\left[(t_k-t_{k-1})<x\right] = 1-e^{-\mu(x-\tau)} \quad ; \tau \leq x < q+\tau$$
$$= 1 \quad ; x \geq q+\tau$$
$$= 0 \quad ; x < \tau$$

The mean of the distribution (C4)

$$m_s = \frac{1}{\mu}(1-e^{-\mu q}) + \tau$$

P_{ij} [j programes in system at t_k given that at t_{k-1} there were i programes in system].

Assuming arrival process as poisson and arrival rate = λ and denote $p(n \mid t)$ as the probablity of n arrival in interval t, we are also assuming arrival is also time homogeneous.

Transition probablities can be shown to be according to (C4):

$$p_{ij} = \begin{cases} 0 \quad , \quad j < i-1 \\ \int_0^q p(0 \mid t+\tau)\, \mu e^{-\mu t}\, dt \quad , \quad j = i-1 \geq 0 \\ p(j-1 \mid q+\tau) e^{-\mu q} + \int_0^q p(j-i+1 \mid t+\tau) \mu e^{-\mu t} dt \quad , \quad j \geq i \geq 1 \end{cases}$$

$$\pi_j = \sum_{i=0}^{x} p_{ij}\, \pi_i$$

To evaluate the limiting probablities of π_j we have to use generating function techniques (C4).

$\bar{n}$= average number of jobs in queue.

$$\bar{n} = \frac{(1-\pi_0) - \beta''(1)/2}{\pi_0}$$

$$= (1-\rho)(1-e^{-\mu q}) - \lambda\tau \quad , \quad \rho = \frac{\lambda}{\mu}$$

$$\beta''(1) = 2\lambda q \delta + 2\delta^2(1-\rho^2) + 2\rho\delta(\rho-\lambda q) - (\lambda\tau+\delta)[\lambda\tau + 2\rho(1-\rho)]$$

$$\delta = e^{-\mu q}$$

To determine average wait time of a job when the markov system is in equilibrium we assume an epoch time and a job arrives , let the service time for this job be s, and W_s is the mean wait time for this job in queue.

If we can choose m such that $0 \leq mq-s < q$

Then, referring to (C4):

$$W_s = \frac{m_s}{1-\alpha} \left[m\lambda(q+\tau) + \left(\bar{n} - \frac{\lambda(q+\tau)}{1-\alpha} \right)(1-\alpha^m) \right]$$

$$m_s = \frac{1}{\mu}\left(1-e^{-\mu q}\right) + \tau$$

$$\alpha = \delta + \lambda m_s \quad .$$

3.5 Conclusion on analytical model building.

Two types of analytical models are discussed in this paper which are very common in multiprogramming system analysis and have been used before to simplify the analysis.

To apply the first model, one needs to collect data on actual system to evaluate parameter $\lambda_1, \lambda_2, \lambda_3 \ldots \lambda_n$ and $\mu_1, \mu_2, \quad \mu_n$ which are arrival rate and service time rate of different priority jobs.

As an example taking CDC 6200 system such data are already collected in day files. What is required is the regression analysis of these data to the assumed distribution and find the goodness of fit. A criterion such as:

$$R^2 = 1 - \frac{\sum_{\mu=1}^{n} T_\mu^2}{\sum_{\mu=1}^{n} (T_\mu - T_s)^2}$$

could be chosen for fitting service time distribution parameters, where:

γ_μ = residuals between theoretical distribution points and actual data.

T_s = average service time.
T_μ = sample service time average.

for best fit $R^2 = 1$, for worst fit $R^2 = 0$.

The second type of model ie., markov model provide the effect of quantum time effect, program swapping time on the system, job queue length, and average wait time for a job. The analysis does not take into account priority scheduling for jobs.

The limitations of the analytical model is the assumption that required to be made for a close solution of very complex problems. The validity of these assumptions is questioned by computer system engineers (G1). But author's view is that the data from the analytical model should be chosen for a guide to use in simulation models. There is also another advantage of analytical model, because during a system design we are always making decisions on hidden assumptions, and analytical model present the problem in a precise mathematical way which brings these assumptions very clearly in view.

CHAPTER 4

IDENTIFICATION OF PARAMETERS AFFECTING THE MULTIPROGRAMMING SYSTEMS.

4.1. Parameters identification in Kronos.

The primary purpose of multiprogramming system is to balance the load on I/O facilities and the processor, so that, both type of facilities are optimally utilised. The primary function of the job scheduler is to bring about this balance by loading a jobset with elements of I/O bound jobs and compute bound jobs.

The most difficult part of the process is the determination of job characteristics. The character of a job changes dramatically during its progress through the system. Hence, a dynamic monitoring facilities for job profile is required, and the scheduling control needs to be probablistics.

To determine the parameters suspected of influencing the system performance, we divide the parameters into two classes. Firstly, we consider the parameters which are physical resources of the system namely central memory, mass storage, and number of Cpus. Associated with each of these parameters there are number of physical limitations, eg., Cpu could only execute at some speed, data transfer between various I/O facilities, mass storage, peripheral processors and central memory could only function at some speed. There is also a physical limitation on the size of the central memory. Thus some of the parameters we may not be able to control or able to control with restrictions.

Second type of parameters are combinations of system software decisions and job characteristics. Some of these parameters we may change and some we do not want to change because our system is a service system and it should provide service at reasonable time and cost.

Thus the following parameters suspected of affecting system throughput are listed:

(a) Central memory size.

(b) Disk(mass storage) access time(seek, rotational and Word transfer.)

(c) Number of CPUs in concurrent operations.

(d) Switching delay between programmes.

(e) Job characteristics.

(f) Job arrival time and Service time distributions.

(g) Job mix.

(h) Main memory management.

(i) Job movement strategy.

(j) Autoroll.

(k) Control point and CPU utilisations.

(l) Job priority management strategy.

(m) Job scheduler frequency of control.

(n) Time slice variable or fixed.

The parameters (a), (b), and (c) are determined and simulated system response to the changes in these parameters are provided in Chapter 5. Chapter 5 also discusses (e), (f). In Chapter 2 a description of KRONOS is given, there the parameters (i), (j), (k), (l), (m), and (n) are described in detail. The analysis that follows in this Chapter concerns Job mix and Memory management applicable to Kronos or similar operating systems.

4.2 Job mix.

This is an important criterion. The creation of a job set for optimum system performance needs, for each job in the set, complete knowledge of job steps, processes(tasks) for each job in the set. A job's demands on system resources are dynamic, hence, the scheduler should try to form an active-job set whose demands on system resources are such that the full use of the system is carried out. To carry out such a policy a mechanism is needed in the operating systems, which collects data on jobs' dynamic nature.

At the moment no such profile data are evaluated, but an improvement in system performance is anticipated once such a step is taken.

All scheduling methods give higher priority to I/O bound jobs than to compute bound jobs, the simple rule is to give the cpu to the job which will release it earliest. The worst scheduling strategy is to give the cpu to a job which computes for longest time before issuing an I/O request. The best scheduling discipline was also found to be preemptive. Under a round robin scheduling scheme, Sherman, Baskett III and Browne (S2) varied the quantum time and found the percentage increase of throughput. Sherman and et.al.(S2) showed that there is an optimum value for throughput for an optimum setting of quantum time.

As discussed before , nature of a job profile is required to be determined before we could predict a satisfactory job mix. Again considering Sherman, BaskettIII and Browne (S2), if X_{n-1} is the (n-1) st. cpu service time for a job and $\hat{X}_{n-1}$ is the (n-1)st. prediction for that job then the prediction of X_n is:

$\hat{X}_n = \alpha X_{n-1} + (1-\alpha) \hat{X}_{n-1}$ where α is a real number between 0 and 1. The larger the value of α the more heavily weighted is the most recent past and the less heavily weighted is the more distant past.

The complete history method predicts that the next cpu service time will be equal to the mean of all past service times for that job. The formula is

$$\hat{X}_n = (X_{n-1} + \hat{X}_{n-1}(n-1))/n$$

These methods we discussed here could easily be incorporated in the system as a control program and operate at a very low priority at some convenient interval. This is going to increase the cpu overhead very little, but would improve the throughput substantially.

4.3 Memory management.

Kronos operates in autoroll system ie., when a higher priority job in memory is rolled out in mass storage, the core space freed by such a procedure may not be fully utilised by the job causing the rollout and the result is the inefficient use of resources.

Some kind of 'demand paging co[illegible] (D2) based on working set idea seems to be efficien[illegible]agement.

One of the advantage of autoroll system is the efficient use of disk I/O. When a job is rolled out, a mass of data is transferred and instead of writing a block of data ; a complete track is written. Autoroll system could be further improved by using extended core storage system (ECS) as a buffer for rollout systems. ECS is a large fast storage designed to be used for streaming data IN/OUT of memory. A streaming rate of 100ns. for 500K 60 bit words can be achieved.

The advantage is the immediate release of central memory field length for job in demand, and finally, in background, the job is transferred to disk storage. The staging of tape files, ie., to copy tape files on disk storage when a job using tape files is rolled out, could also go through ECS.

CHAPTER 5

PERFORMANCE EVALUATION THROUGH SYSTEM SIMULATION. DEVELOPMENT BUILDING AND TESTING OF MULTIPROGRAMMING SYSTEM MODEL.

5.1. Introduction.

A model of a simplified Kronos system has been developed and the performance problem is studied with the help of this model.

The simulation problem is approached in four phases. Phase 1 consists of building of the model. Phase 2 is the implementation of the model using simulation language called SIMSCRIPT. Phase 3 is the study of the system using the model under various resource parameters. Phase 4 is the analysis of the results obtained from the model.

5.2. Objective of the model development.

The objective of the system model development is twofold. The first objective is to help the system designer with the results obtained for better insight into the system; the second is to understand an existing system behavior under varying resources and job input conditions, ie., performance measuring.

The objective of this work falls under second category as mentioned above. The system is already designed and working. The objective is to find the system behavior

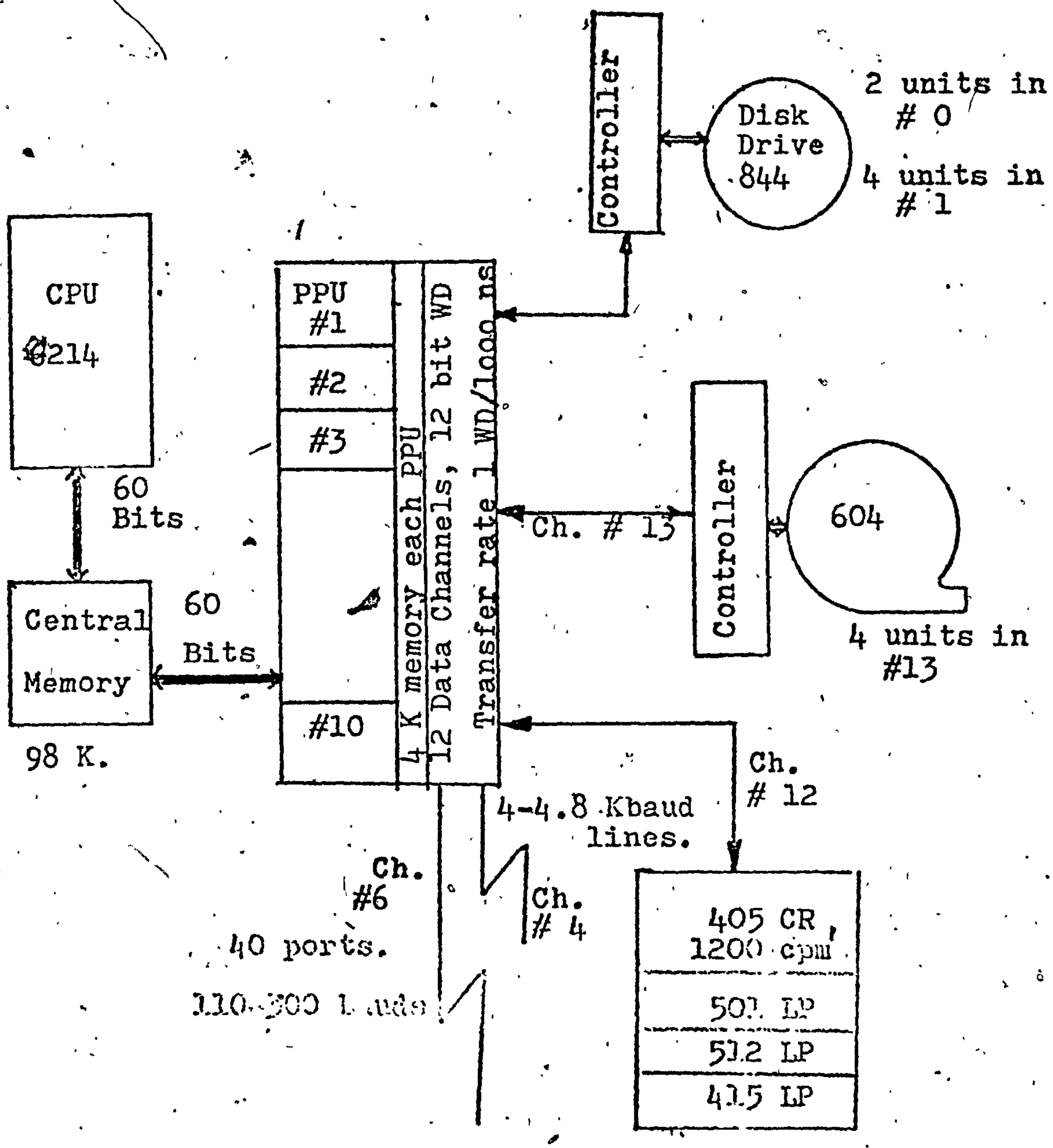


Figure 5-1. System Configuration.

under different resource conditions that are below and above the present working level of operations. The final goal is to uncover the critical and noncritical resources of the system affecting the performance of the system. Associated with the resources there are system parameters, such as, CPU scheduling with time-slice, and job arrival rate. The effects of these parameters are also intended to be uncovered.

5.2.1. Output from the model

The goal of the simulation is to obtain some measures of the performance of the system. The output from the model provides such measures.

Measurement is divided into two distinct parts. The first part is the measurement which an user is interested in knowing (i.e., Throughput). The second part consists of a set of values that a system designer or a system evaluator may want to know for optimum system design or system operation. The second part includes information pertaining mainly to system queues and few other items such as average central memory usage, percent CPU time.

A typical output from simulator is shown in figure 5-3.

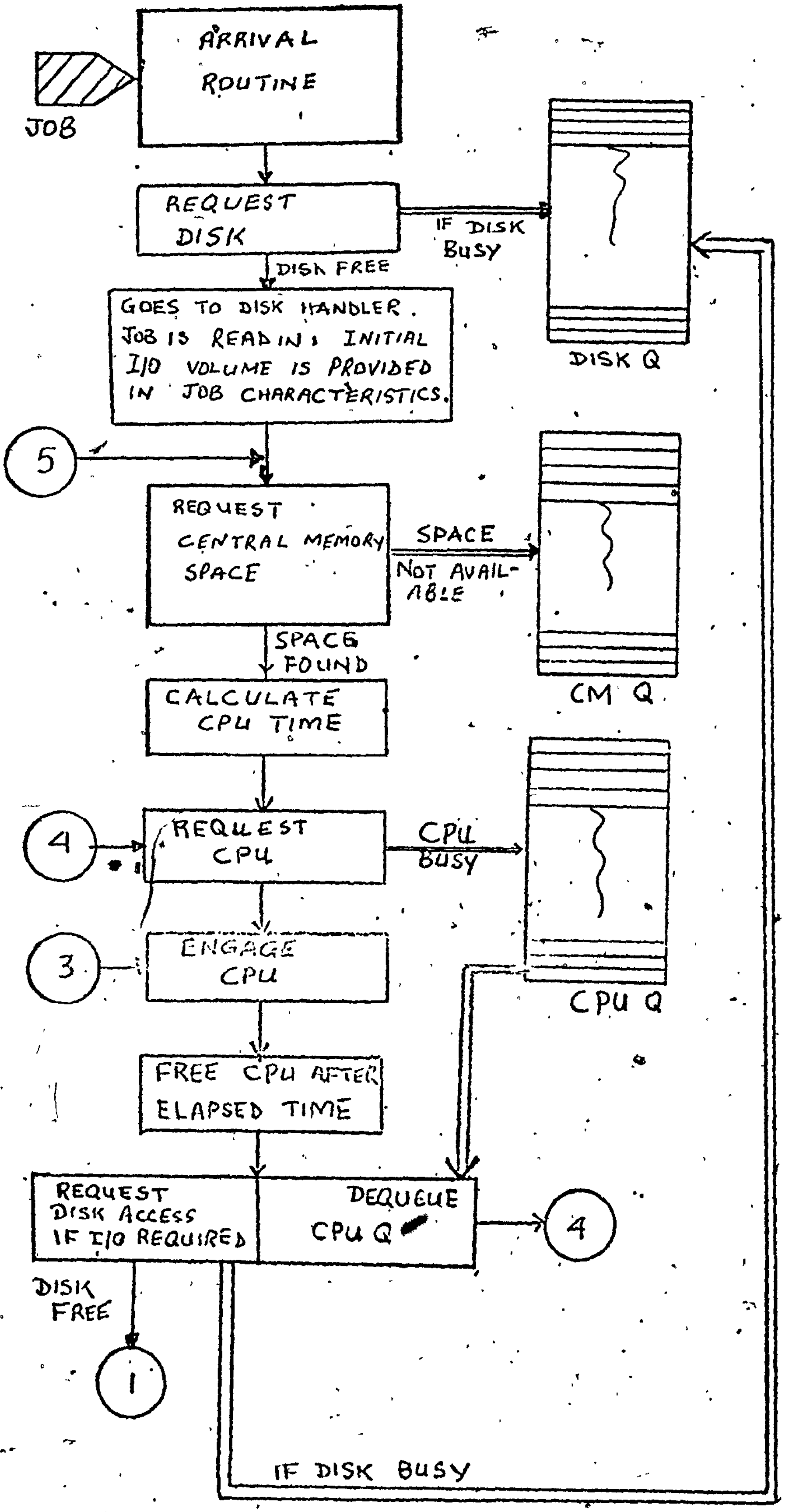

JOB
ARRIVAL ROUTINE
REQUEST DISK
IF DISK BUSY
DISK Q
DISK FREE
GOES TO DISK HANDLER. JOB IS READ IN; INITIAL I/O VOLUME IS PROVIDED IN JOB CHARACTERISTICS.
5
REQUEST CENTRAL MEMORY SPACE
SPACE NOT AVAILABLE
CM Q
SPACE FOUND
CALCULATE CPU TIME
4
REQUEST CPU
CPU BUSY
CPU Q
3
ENGAGE CPU
FREE CPU AFTER ELAPSED TIME
REQUEST DISK ACCESS IF I/O REQUIRED
DEQUEUE CPU Q
4
DISK FREE
1
IF DISK BUSY

Figure 5-2. System Model.

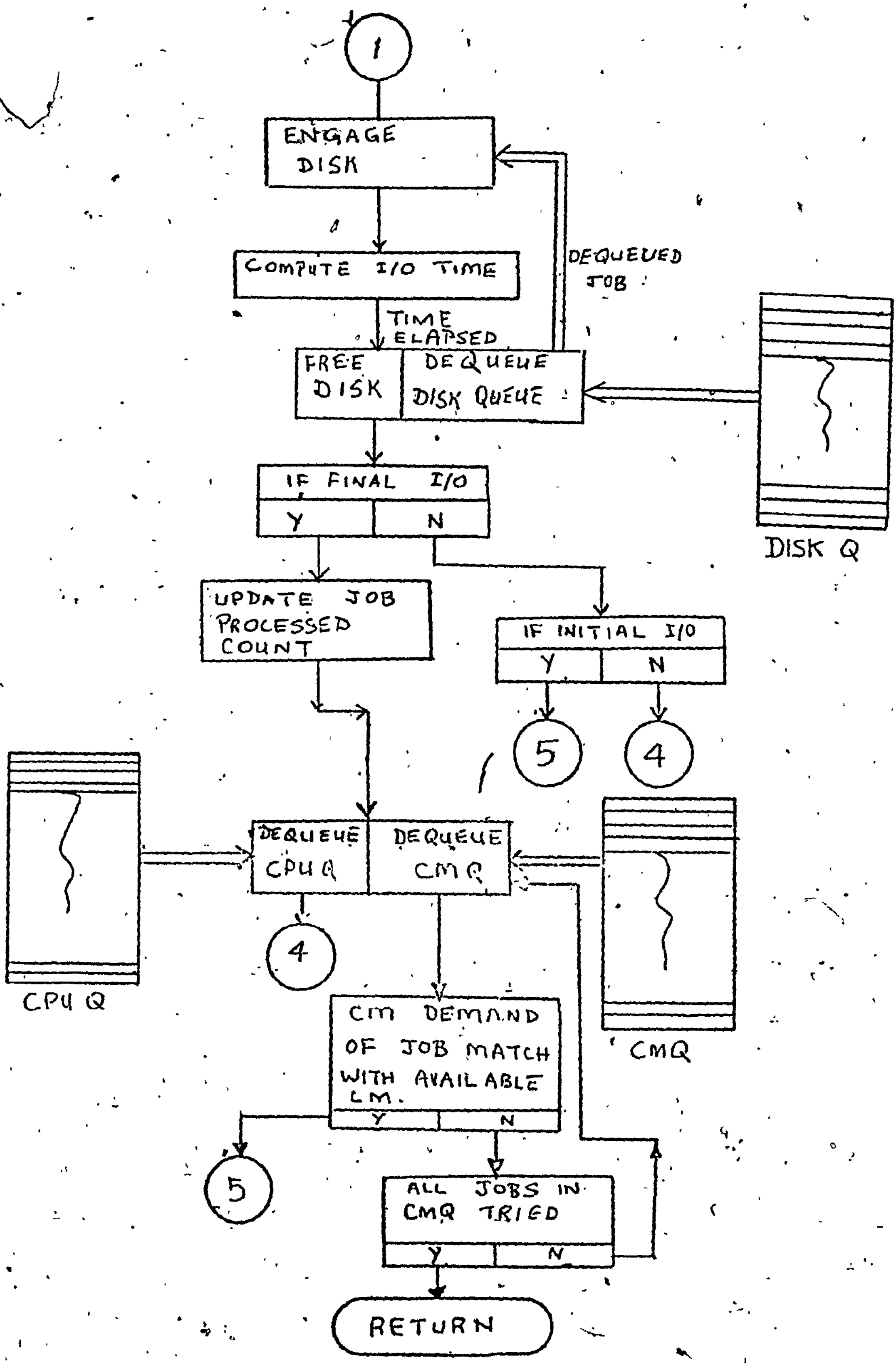
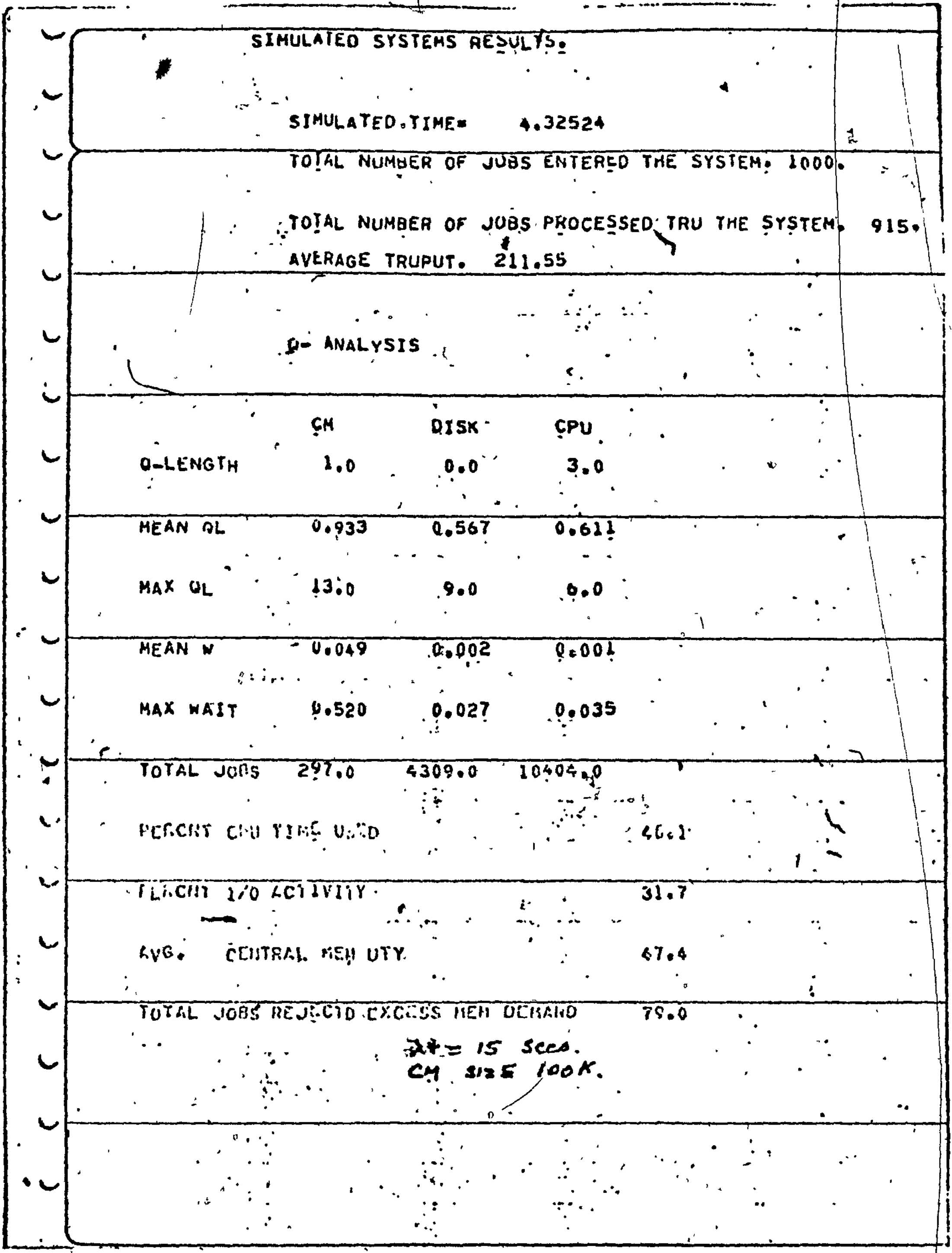


Figure 5-2 (cont.)



```
                 SIMULATED SYSTEMS RESULTS.

                    SIMULATED.TIME=     4.32524
                    TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

                    TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  915.
                    AVERAGE TRUPUT.  211.55

                    Q- ANALYSIS

                       CM        DISK       CPU
      Q-LENGTH         1.0        0.0        3.0

      MEAN QL         0.933      0.567      0.611

      MAX QL          13.0        9.0        6.0

      MEAN W          0.049      0.002      0.001

      MAX WAIT        0.520      0.027      0.035

      TOTAL JOBS    297.0     4309.0    10404.0

      PERCNT CPU TIME USED                            46.1

      PERCNT I/O ACTIVITY                             31.7

      AVG.   CENTRAL MEM UTY                          67.4

      TOTAL JOBS REJECTD EXCESS MEM DEMAND            79.0
```

Δt = 15 secs.
CM size 100K.

Figure 5-3. Output from system model.

5.2.2. Input to Model.

System behavior is influenced by the job characteristics and input rate. Thus, it is crucial to generate the input jobs to the model as realistic as possible. Analytical models could be useful here.

In the present work, a permanent file has been created by a program module which is separate from the modules simulating the system. The file contains one thousand jobs having exponential distributions of arrival times and resources demands. This is used as an exogenous tape in the simulation as a source of jobs.

5.2.3. Conclusion on system model building.

The model development has been divided into three distinct tasks:

Task 1 is the design of system model; Task 2 is the design of the model output ; Task 3 is the design of the model input.

In the present work, the approach to system model building is to follow a job logically through the system. Logical modelling is the modelling of the structure of the system. When a job is traced through the system, the structure of the system is uncovered. Following this approach the modelling problem reduces to series of operations to be performed according to model structure. Such a model is developed here.

5.3. Description of the model.

The system model is shown in figures 5.2 and 5.2A. The model is developed to correspond with job flow. A job arrives at the computer centre and is read into central memory buffer by the peripheral processor. Disk accesses are necessary to read jobs into mass storage from central memory. If the disk is busy, the job is put in disk queue otherwise the job is handed over to disk I/O handler . The disk I/O handler performs the initial disk I/O for the job, and at termination returns to memory assignment module , requesting central memory space, for job execution. If sufficient central memory is available , the job is assigned memory, otherwise it is put in central memory queue.

The job occupying central memory requests CPU service . The CPU is granted to the job if available, otherwise the job is queued in the CPU queue. The CPU provides service to the job for the amount of time indicated in job characteristics.

The total CPU time demanded by the job is distributed equally among I/O activities during execution; the CPU is assigned to the job for time slice that is calculated in ARRVL routine of the Simulator.

This philosophy of distributing the CPU time over complete execution time of the job including I/O provides a balanced load on CPU and DISK. For I/O operation the job is always handed over to the disk I/O handler.

The I/O activity during execution is assumed to be 1 pru unit, and when this is accomplished, the job requests CPU for performing computations. This continues until total time (i.e., compute) demanded by the job has been satisfied. Incidentally satisfying all compute time also satisfies execution time I/O requirements but final I/O operation for results has yet to be performed. Hence, the job is sent to the I/O handler, and having completed the final I/O it is sent to the job termination module.

The job termination phase releases all CM space occupied by the job, updates job count , dequeues both CM and CPU queues, and pushes the waiting jobs through the system. It also generates information regarding central memory usage, disk and cpu utilisations.

5.4. Monte carlo simulation of job stream.

One of the difficult problems in system simulation is the varying nature of the job which is very difficult to predict.

Jobs to computer systems are programs , and behaviors of programs change in a short period of time. A heavy compute bound program becomes I/O bound suddenly.The demand of a job on system resources could fluctuate dramatically. The demands on system resources of a typical job could be categorised into the following list:

(a) Program's demand on central memory.

(b) Program's demand on CPU time.

(c) Program's demand on disk access.

(d) Program's demand on magnetic tape drive.

(e) Program's demand on card reader.

(f) Program's demand on line printer.

(g) Program's demand on communication facilities.

Thus the simulation programs generating the job input stream should not only generate jobs at random intervals reflecting the operating environment but also generate the resource demands on a random basis on all or part of the resources categorised in the list.

5.4.1. Job generator description.

The job generator functions independently from the main simulation model. The program is written in FORTRAN ; it creates jobs, stores them on permanent file which is used as a source of the job stream to system model. The job characteristics, generated by the job generator, are defined by the stochastic variables, and are specified by poisson distribution functions. The validity of the assumption is discussed in 3.2.

In simulation instead of using distribution function, a related function, cumulative distribution function is used which could be defined as $F(x)=\int_{-\infty}^{x} f(x)\ dx$; where f(x) is the distribution function.

F(x) is monotonically increasing and its value is positive ranging from 0 to 1. The value of $F(X_0)$ is the probablity that the value of x is less than or equal to X_0.

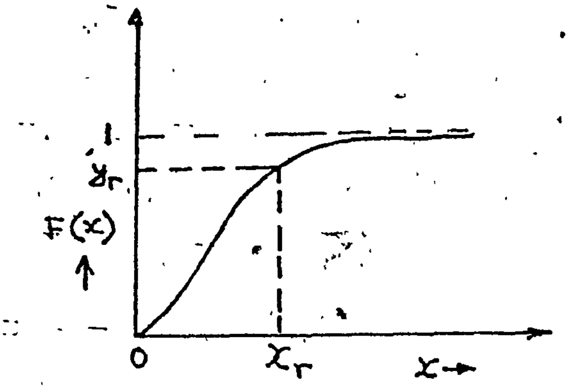


Fig. 5-4A. Cumulative distribution.

Actually what is really needed is the inverse of cumulative distribution function. When simulating the system we need to generate stochastic variables for the attributes of every job that enters the system. For each variable, we need to generate a set of values at random which is part of the distribution.

In most system libraries one program is available for generating uniformly distributed random numbers within the range of 0, 1. Random numbers could also be obtained from Tables and one such table is availabl from Rand Corporation, " A Million Random Digits wit 00,000 Normal Deviates."

But to generate random variables fitting an arbitrary cumulative distribution, we need to find the inverse function. Let the random variable be X_r and cumulative distribution is $F_{X_r}(x_r)$. If y_r is the uniformly distributed random digit, then we can equate :

$$y_r = F_{x_r}(x_r)$$

$$\text{or,} \quad x_r = F^{-1}_{x_r}(y_r)$$

x_r is the desired random variable. This approach will not work if we can not find the inverse cumulative distribution function.

One of the important characteristics of the job is the arrival time. The distribution function for negative exponential distribution is $P_r(\ x \leq t\ ) = 1 - e^{-t/T_A}$.

Let $\lambda = 1/\ T_A$ is the average number of jobs per unit time.

$$F(t) = 1 - e^{-\lambda t}$$

Let p be the uniformly distributed variable as generated by the system built in function.

$$\text{then } p = 1- e^{-\lambda t}$$

$$t = -\ T_A \log_e (1 - p)$$

replacing p_r instead of 1-p

$$t = -\ T_A \log_e p_r \quad \text{-----------}(5.1)$$

Hence job interarrival time is given by (5. 1)

5.4.2. Determination of Mean values of job parameters.

The following data were collected from Kronos daily accounting summery:

(a) Total CPU time in secs. for Batch and Telex origin jobs.

(b) Central memory in kiloword hours for Batch and Telex origin jobs.

(c) Mass storage access in kilo Pru units for Batch and Telex jobs.

(d) Card reader # of cards read in thousands.

(e) Line printer # of lines printed in kilo lines.

(f) Total number of jobs for Batch and Telex origin jobs.

From the above set of data the following mean values are calculated:

(a) Average CPU time = CPU time/job.

(b) Average CM requirements/job.

(c) Average mass storage access in PRUs.

(d) Average number of cards read.

(e) Average number of characters outputted.

5.4.3. Distribution of disk access of a typical job.

The total number of disk accesses during the computation period of the job is determined. Since one disk drive is simulated , disk accesses are proportionately reduced. The PRU accesses are distributed uniformly over the whole period of job as shown in figure 5-4B.

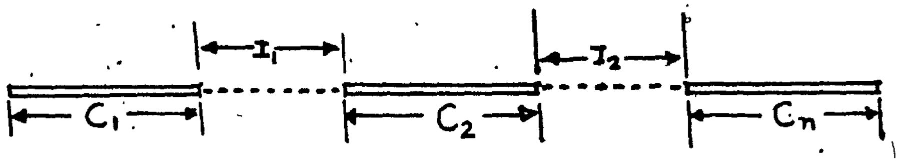


Fig. 5-4B. Distribution of I/O and Compute times.

C_1 , C_2 ....C_n are the compute time bursts of the job which are interspersed with I/O activities of I_1 , I_2 ... ...I_n quanta of time. It is also assumed $C_1=C_2=\ldots=C_n$ and $I_1=I_2$ ....$=I_n$ and each I/O activity consists of 1 PRU unit.

Apart from this it is also assumed that every job would have some disk access at the beginning and at the end when printing the results and for these separate stochastic variables which are known as Initial I/O and Final I/O have been generated.

These variables have negative exponential distributions as their density functions. The mean values are determined from Kronos accounting summary.

The averages are also computed for card reader and line printer activities.

The Central memory buffer for I/O is taken as 640 characters long, (ie., 1 PRU unit.)

5.5. Modelling of queues in the system.

There are three queues in the system, central memory queue, disk queue and cpu queue. All these are organised in similar manner. The queue discipline is First in and First out (FIFO).

The queue size is unlimited and is only limited by the available storage. Each queue has various pointers of which queue and dequeue pointers are defined in simulator. Since simscript is used to implement the model the problem of queue handling has been reduced to a large extent. For FIFO organised queues in Simscript one needs two pointers called first and last pointers. The pointers are defined by having F and L in front of the Q-name. The first and last pointers for central memory queue are called FCMQ and LCMQ where CMQ is the name of central memory queue as defined in the Simscript defination form.

Two commands are extensively used in all queue management routines and thay are "FILE" and "REMOVE" for queuing and removing a job from queue.

The queue management in simulation is done through a series of Queue and Dequeue subroutines. The various parameters which are of interest and are affecting the system performance implicitly, are monitored in the simulation.

5.5.1. Central memory queue management.

To manage queuing and dequeuing a job in central memory queue, two algorithms are used and they are as follows:

Insert entry in queue.

1. File job pointer to head of queue.
2. $\sum TQ + (Time - T_{last})\ Q \rightarrow \sum TQ$
3. $Q + 1 \rightarrow Q$
4. $Max\ [Q,\ QMax] \rightarrow QMax$
5. $N + 1 \rightarrow N$
6. $Time \rightarrow T_{last}$

Remove entry from queue.

1. Remove the job pointer from tail of the queue.
2. $\sum TQ + (\ TIME - T_{last})\ Q \rightarrow \sum TQ$.
3. $Q - 1 \rightarrow Q$
4. $\sum TW + (TIME - T_{in}) \rightarrow \sum TW$
5. $Max\ [\ W_{max}\ ,\ TIME - T_{in}] \rightarrow W_{max}$
6. $TIME \rightarrow T_{last}$

The figure 5-5 shows the queue structure and the notations are explained in the following paragraph.

W_{max} : Maximum waiting time.

$\sum TW$: Waiting time accumulator.

Q_{max} : Maximum queue length.

Q: Current queue length.

$\sum TQ$: Length X Time product accumulator.

T_{last} : Time of last entry / removal.

T_{in} : Time of entry in queue.

N: entry count.

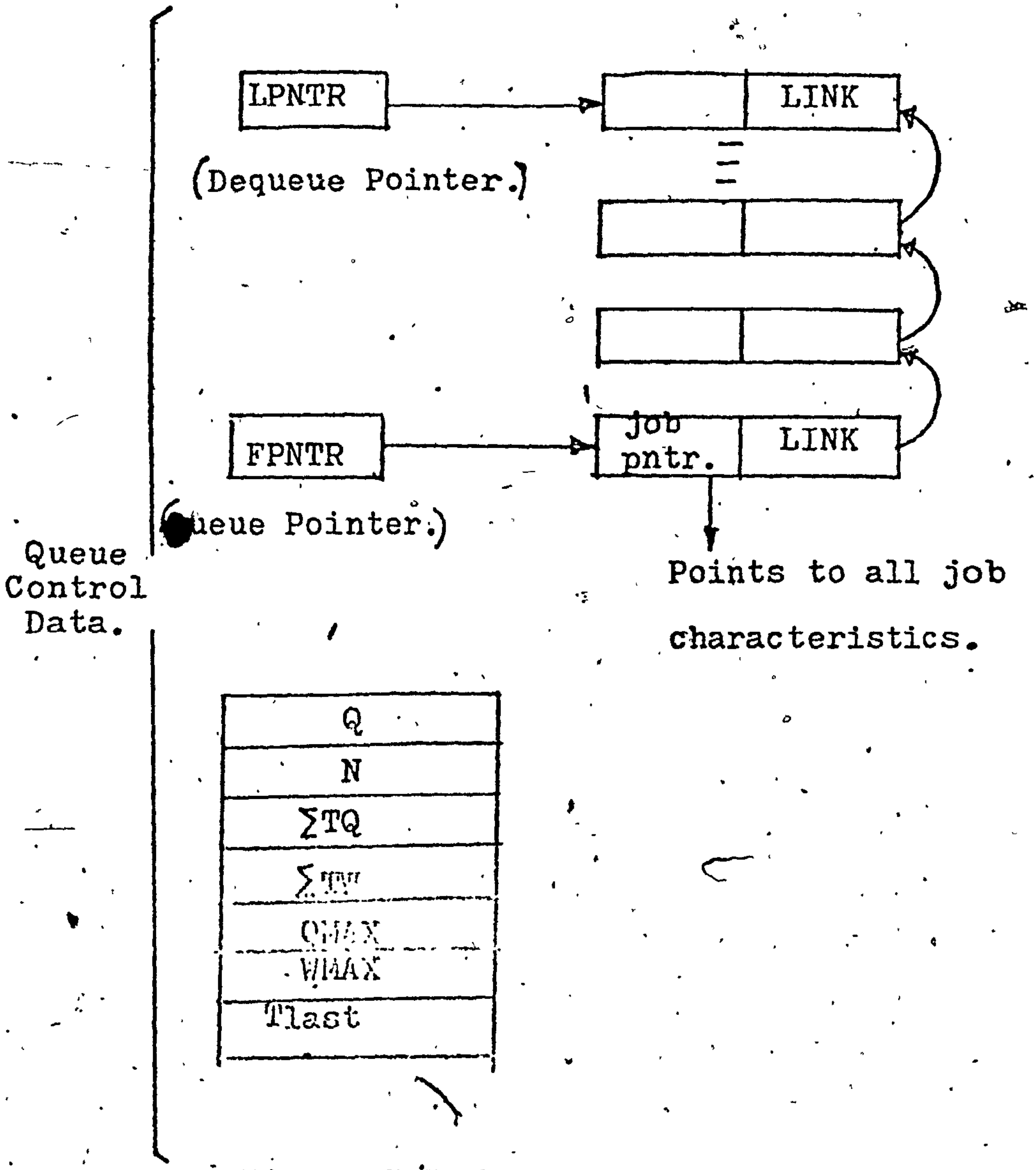


Figure 5-5 Queue Structure.

5.5.2. CPU and DISK queues.

These queues are handled in the same manner as the case of central memory queue. All the queue parameters as shown in figure 5-5 are also provided in the management of the CPU and DISK queues. The only difference is that the parameters have different names.

5.5.3. Conclusion on queue management.

The simulator has three main queues, central memory , CPU and DISK . A job, depending on situation, could reside in any one of these queues. All queues are managed in a similar manner and queuing and dequeuing philosophy is First in and First out. Simscript has provided excellent feature on queue management. Queuing and dequeuing routines include algorithms for system intrinsic parameters that could be useful to system evaluator.

The queue sizes are only limited by operating system storage capacity and there is enough storage for thousand jobs that are used for simulation.

5.6. Simscript implementation of the model.

The model as shown in figure 5-2 is implemented as simulator using simscript language. The simulator is broken down into a set of routines each performing some specific functions. These routines are again classified into two groups (1) Endogenous (2) Exogenous routines. Referring to figure 5-6, there are six endogenous routines called PHAS1, PHAS2, PHAS3, PHAS4, PHAS5, and PHAS6 and three endogenous routines called ANALYZ, ARRVL and ENDSIM. Exogenous routines are caused by external events and endogenous routines are caused by routines running in the systems.

Any job arriving in the system causes the exogenous event ARRVL which in turn causes PHAS4 with job Input flag set. PHAS4 engages disk and perform Disk I/O and on termination causes PHAS5. PHAS5 causes PHAS1 because the job is just read in and requires central memory for execution. PHAS1 requests central memory space and having got the space, it causes PHAS2 which in turn causes PHAS3 and eventually PHAS4 and PHAS5. Thus the job after being read by PHAS4 is processed sequentially through PHAS2, PHAS3, PHAS4 and PHAS5 as shown in figure 5-6. PHAS5 returns the job to PHAS2 if it has still some computation to perform. The job cycles through PHAS2, PHAS3, PHAS4, and PHAS5; this continues until the compute time required by the job has been completely satisfied and the final I/O activity to produce results has been performed. Final exit of the job is performed through PHAS6 routine which releases all shared facilities.

As shown in figure 5-6, the system data and queues are declared in simscript defination form and all routines in the system access and modify these set of system data.

The exogenous ARRVL routine is triggered by the incoming jobs. All incoming jobs are simulated and created by the offline job generator and resides in permanent file and the file is used as a exogenous tape to simulator. This is a brief description of simulator of fig. 5-6, the description of individual routines follows:

5.6.1. ARRVL routine.

The main function of this routine is to create a temporary entity JOB. It also reads job parameters such as CPU time, Central memory space, Initial disk access required by the job, Final disk access required by the job and number of I/O record transfer during computation phase of the job in central memory. It also distributes the CPU time equally over the whole period of job's resident in central memory. This creates equal pockets of compute time interspersed with I/O record transfers. It sets the input flag and creates and causes the endogenous event PHAS4.

5.6.2. ANALYZ routine.

This is an exogenous routine caused by the permanent file used as a source of jobs. This is caused by the external event at termination of all jobs. The main function of the ANALYZ is to provide hardcopy output of results sought from

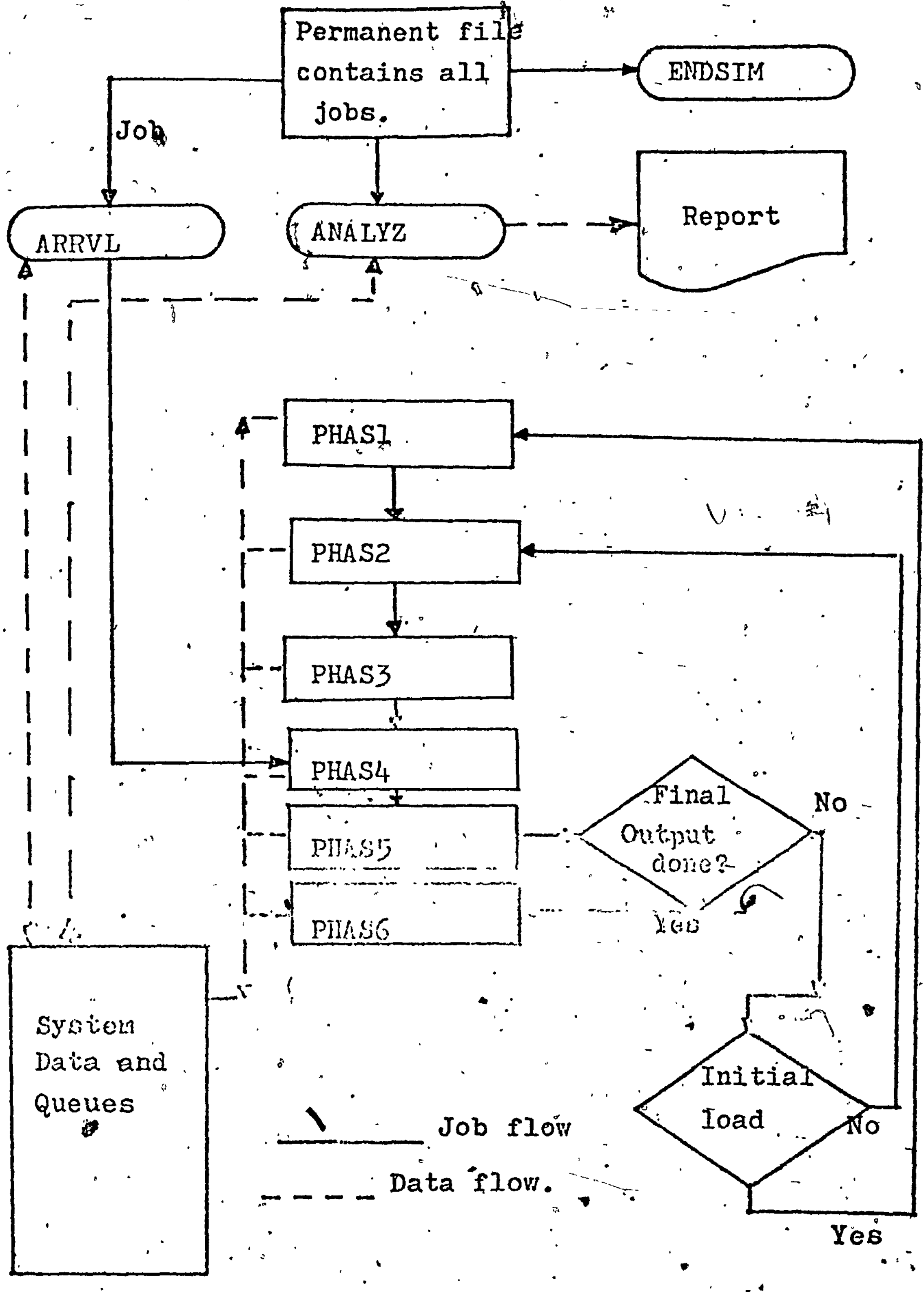


Figure 5-6. Simulator program interaction.

simulator. It provides information regarding simulated time, average throughput, other queue parameters required by the system designer or evaluator and estimates of system resource utilisations.

5.6.3. ENDSIM routine.

This routine terminates the simulation process and in some special cases terminates the present simulation process and read in a new set of data initialisation tape and commences a simulation run.

5.6.4. PHAS1 routine.

The main function of this routine is to load the job into central memory if central memory space is available and in the event of insufficient central memory space it loads the job in the central memory queue. If the job is successfully loaded in the central memory then it creates and causes PHAS2 routine.

5.6.5. PHAS2 routine.

The main function of this routine is to get CPU service for the job. If CPU is occupied then the job is put in CPU queue , otherwise CPU is assigned to the job and the routine exit after creating PHAS3. Having completed all the CPU time activity the routine could exit creating PHAS4.

5.6.6. PHAS3 routine.

The main function of this routine is to cause job records' input /output activities between compute activity. The routine exits after creating PHAS4.

5.6.7. PHAS4 routine.

The main function of this routine is to perform disk I/O as required by the job. If the disk drive is busy the job is put in disk queue, otherwise disk is engaged and I/O is performed. In simulation model for Kronos, I/O is performed in three different stages, a set of I/O activity during job's residency in central memory a a set of I/O activity during job's final result development and the I/O perfomed during loading of job. All these different I/O activities are recognised by this routine and executed. This routine exits after creating PHAS5.

5.6.8. PHAS5 routine.

This routine is executed after an I/O activity is performed by the job. This routine exits creating one of these routines PHAS1, PHAS2 and PHAS6. If initial job loading is done then PHAS1 is created, if I/O is performed during job's residency in central memory then PHAS2 is created and if final output is over then PHAS6 is created.

5.6.9. PHAS6 routine.

This is the terminating routine of the simulator. All jobs are processed through this/routine at their termination. The important functions of this routine is to free the shared resources of the system. The shared resources occupied by the completed job are central memory and CPU. This routine dequeues waiting jobs from central memory and CPU queues and initiates them.

The routine also performs some calculations regarding memory, CPU and DISK utilisations. The calculations for CPU and Disk utilisations are keeping the running total for CPU time used and I/O activities performed by each job.

The following procedure is followed to compute %CM utilisation.

Let τ is the time a job j is allocated central memory of ΔC. Let the job finishes at time t,then we may say $\Delta C(t-\tau)$ is a measure of cetral memory utilisation.

$\sum_{n=1}^{\text{Total number of jobs}} \Delta C_n (t_n - \tau_n)$ gives a total measure of central memory utilisation.

Then mean central memory utilisation

$$= \frac{\sum_{n=1}^{K} \Delta C_n (t_n - \tau_n)}{T}$$

Where, K: Total number of jobs.
T: Total simulated time.

The routine exits after creating PHAS2 event for a job and PHAS1 for another job if needed.

5.7. Simulation Results.

This is the third phase of the work regarding simulation approach to the problem. Here a simulator is used to simulate Kronos operating system under the following restrictions:

(a) No autoroll mechanism.

(b) Simplified priority scheme of FIFO principle.

(c) Control points are eliminated.

(d) Reduced number of disk drives.

(e) Multiterminal, Telex and Batch jobs are all grouped into one class with sample means of arrival rate, and various resource demands. The resource demands are formulated with probablistics notion and assumed to follow poisson density distribution.

The plan for using the model to find the interaction and the effect of different system parameters has been organised in the following way.

(a) First we try to find the system behavior under normal operating conditions, ie., under existing facilities and workloads.

(b) The next stage was to vary the different system resouces such as central memory size and average disk access time. The effect of these changes on system throughput and intrinsic queue parameters are noted and evaluated.

(3) In the last phase, CPU time slicing is introduced and the effect of this parameter on system is observed. Our first objective needs some clarification regarding normal setting for operation.

The normal operating conditions are as follows:

(1) Central memory capacity 300K octal.

(2) 841 disk drive with 40 ms. average seek time, rotational delay of 20 ms. and byte transfer rate of 125000 bytes/sec.

(3) One cpu.

(4) Apart from system resources, the job parameters which are considered as stochastic variables, have poisson distributions. Hence norms for CPU time demand, central memory demand and I/O activities at the job's entry time, runtime, and termination time are evaluated from job's accounting file to generate the required distributions.

Having decided on what different set of experiments we want to do with our model, the next part of the plan was to decide what to measure to determine the system behavior. This is a very delicate question to decide. Hence the measurement is split into two categories, first category is the measurement which gives us the direct evaluation of the system namely two things (a) Throughput (b) Mean Wait time.

(a) throughput is simply defined as the number of jobs processed per hour.

(b) Mean Wait time: This is defined in our system as the mean waiting time in system queues. There are three queues, CM, DISK, and CPU in the system. During simulation run, the mean waiting times in each of these three queues were recorded and the Mean Waiting Time has been defined as the largest of these three values.

A number of variables are associated with resource management. These variables are part of three shared resources of the system (a) Central memory (b) Disk and (c)CPU. They include the following (1) Mean, Maximum, and Running values of Queue length of jobs in queues of CM, DISK and CPU.
(2) Mean, Maximum and running values of Waiting time for jobs in CM, DISK, and CPU queues.
(3) Total time used by CPU, DISK, and number of jobs rejected due to excessive CM demand and average central memory utilised.

5.7.1. Simulation run setup.

The model is coded in simscript and load module is created as shown in figure 5-7. The load module is stored in permanent file called FINSIM.

The next step is to set up the job deck for simulation run with different initialisation decks. This is achieved as shown in figure 5-8. Tape 14 is the permanent file created as a source of jobs to the system model. This file contains one thousand jobs with mean inter arrival time, Cpu time demand, central memory demand and I/O activities. The jobs generated reflect true jobs in computer centre.

The several simulation runs with different initialised values of system variables are achieved through system specification card. A system specification card must be provided with each initialisation deck. The set up of the system specification card is shown in table 5-1.

As shown in table 5-1 a 1 punched in column 25 automatically triggers a new simulation run with new initialised values of system variables as specified in the initialisation deck. An initialisation deck always terminates with blank card. Column 25 should be left blank in the last system specification card. The field 37-42 is meant for exogenous event tape 14 which is used as a source of jobs. There are other fields available in the system specification

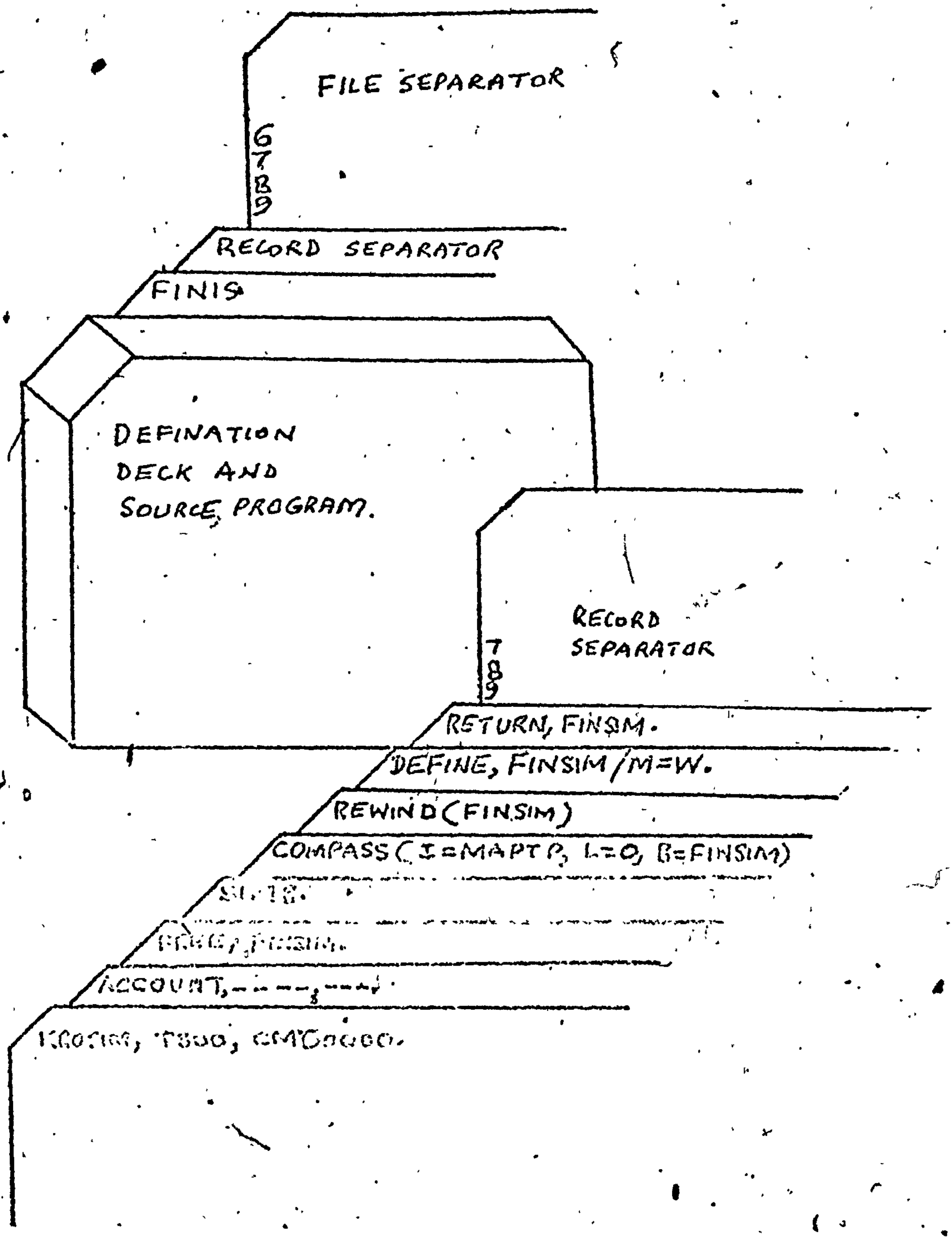


Figure 5-7. Load module generation of system mbdel.

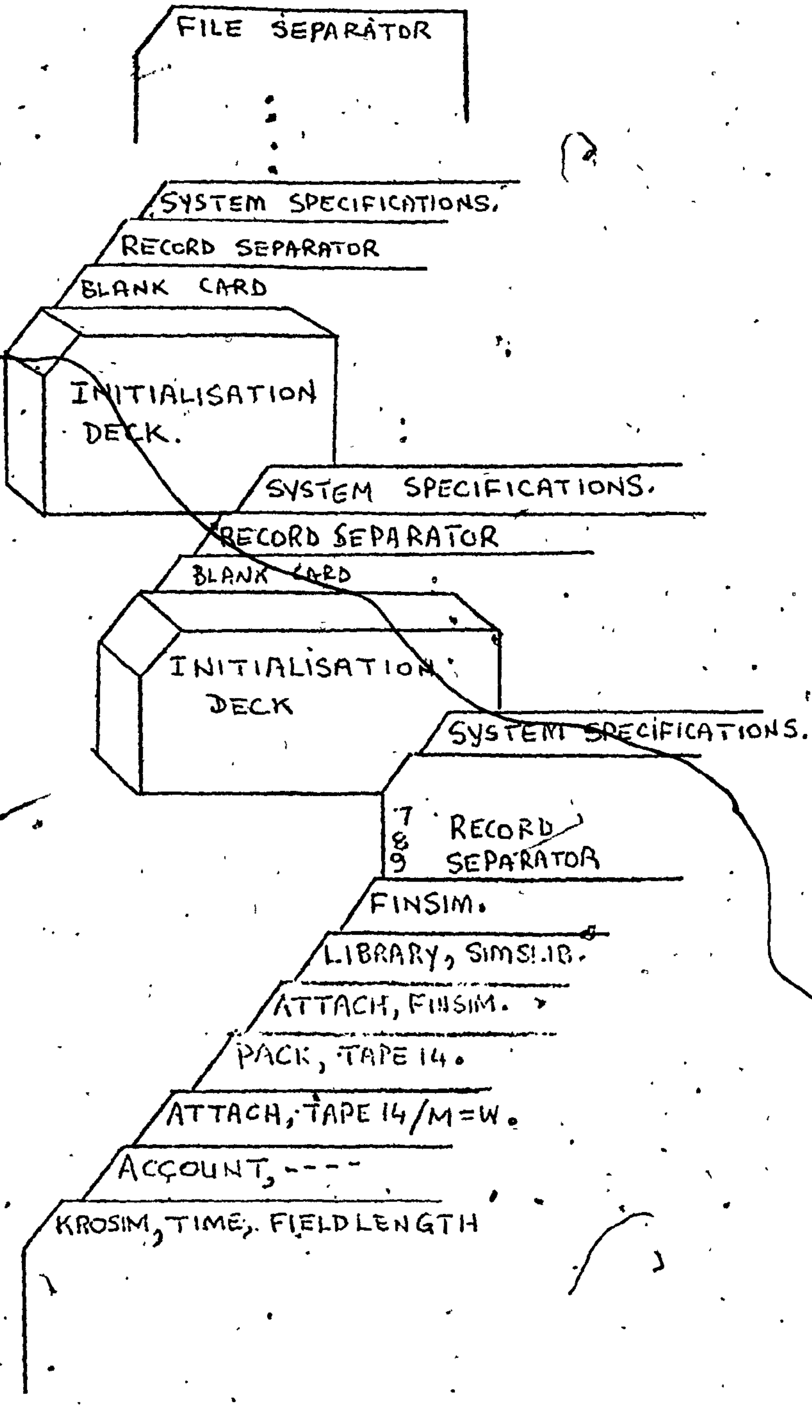


Figure 5-8. Simulation run.

TABLE 5-1. System specification card.

COLUMN.	VALUE PUNCHED.	COMMENTS.
1	1	Card identification.
2	1	lists initialisation cards on output.
7-12	79	Highest array number.
13-18	100	Minutes per hour.
19-24	1000	Hours per day.
25	1	For initialisation after STOP statement for another simulation run.
37-42	14	Exogenous event tape, source of jobs.

card providing number of lines per page, file number for initial condition deck and file number for report tape which are not discussed here since they are left blank. They assume a standard value by default.

5.8. Results and Interpretation.

The results obtained from the model and their possible explanations are presented in the following order:

(1) Throughput Vs Mean rate of job arrival.

(2) Throughput Vs Central memory size.

(a) For mean rate of job arrival = 4 jobs/min.

(b) For mean rate of job arrival = 60/14 jobs/min.

(3) Throughput Vs Mean disk access time.

(4) Throughput Vs Cpu time slice.

(5) Mean Wait Vs Central memory size.

(a) mean time between job arrival = 15 secs.

(b) mean time between job arrival = 14 secs.

5.8.1. Throughput Vs Mean rate of job arrival.

The results obtained from simulation is shown in figure 5-9. The mean arrival rate is changed from 60 secs., ie., 1 job/min. to 6 secs., ie., 10 jobs/min. and it is observed that the throughput is a linear function of arrival rate within a region of 1 to 5 jobs/min. mean arrival rate. This is due to the fact that the system is not saturated and services all jobs without delays. The resources are adequate in this region with the result the throughput matches with the mean arrival rate.

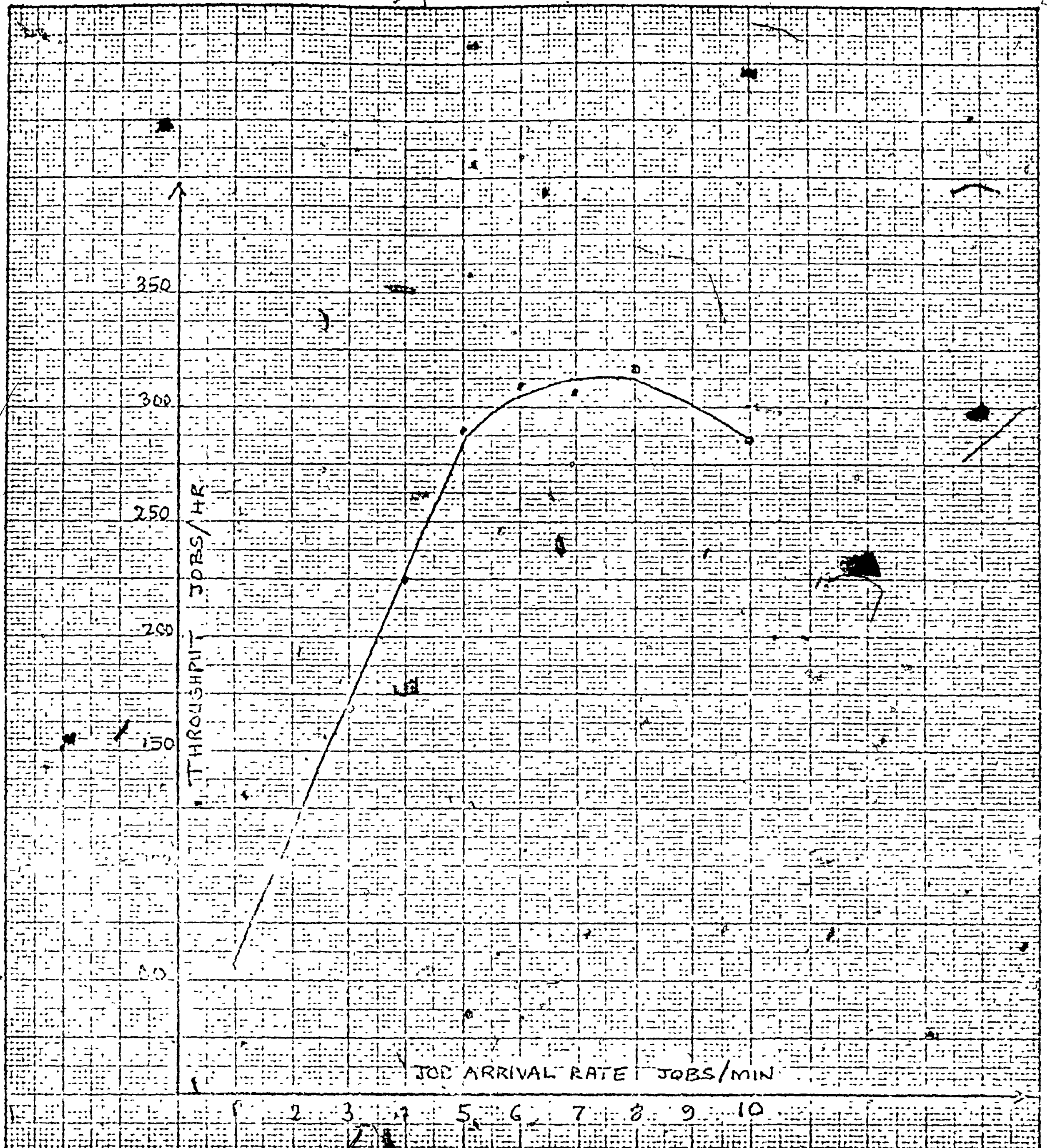


Figure 5-9. Throughput Vs Job Arrival Rate.

10 MM/CM

When the mean arrival rate is changed from 5 to 10 jobs/min. the system exhibits tendency to saturation. The throughput keeps on rising and reaches the maximum value of 315 jobs/min.

The loss of throughput is caused by resource limitations. The resources are tied up by the jobs which are scheduled in central memory and new jobs arriving in the system are,therefore, queued in central memory queue. In later part of the experimentation, it would be shown how an increase in resources increases throughput.

In conclusion, it is established that the throughput of a system depends to a large degree on mean rate of jobs which are serviced by the system. It is uncovered that the throughput is directly proportional to mean arrival rate,for job arrival rates between 1 to 5 jobs per min., as shown in figure 5-9.

5.8.2. Throughput Vs Central Memory size.

Central memory is an important resource in this system. This could be evident from the results shown in figure 5-10. Referring to appendix A for system results, it is evident that the loss in throughput is mainly caused by rejection of jobs due to excessive memory demand (, ie., the memory demand exceeding total system memory). However, this is not the only factor, a lower CM capacity results in higher number of jobs in CM queue waiting for service; This is the reason for

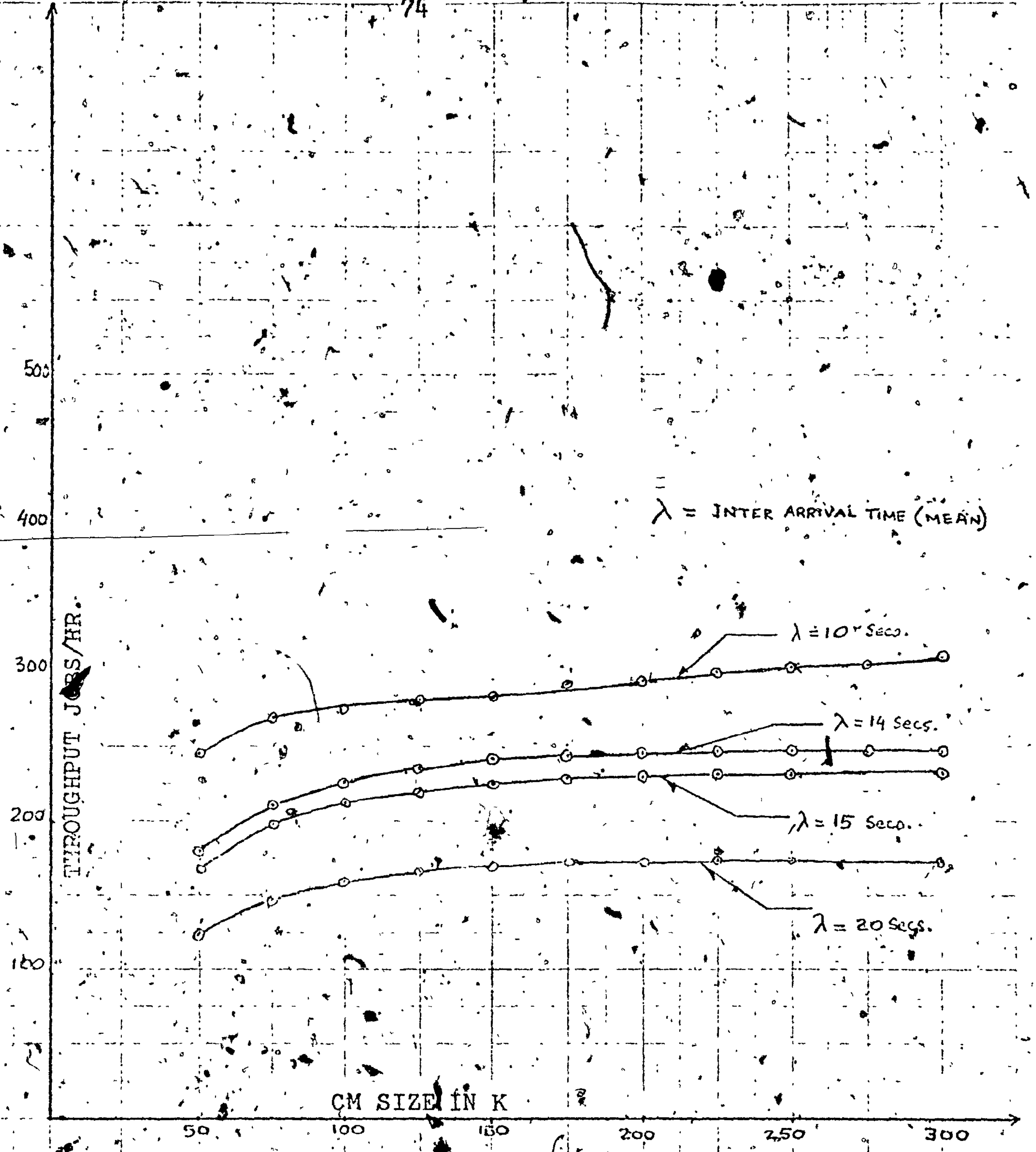


Figure 5-10. Throughput Vs. CM Size.

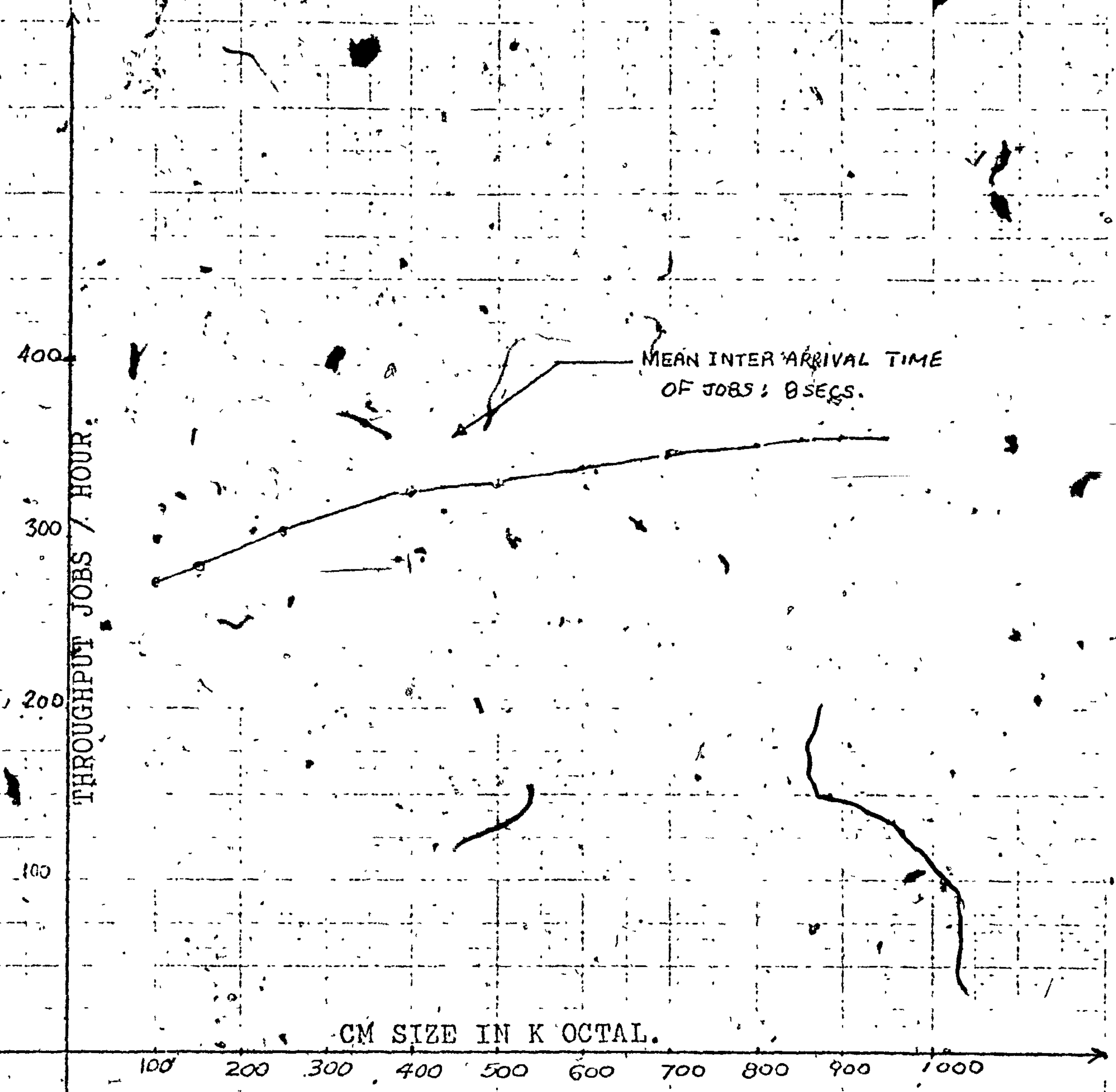


Figure 5-11. Throughput Vs. Central Memory Size.

20 Squares to the Inch

lower throughput. This could be supported from figure 5-11 where CM size is already 300K ; the system results support that, at 300K CM size, there are no rejection of jobs. In figure 5-11, it is shown that the throughput is increasing with the increasing CM size; this is happening when the system is saturated as shown in figure 5-9.

Thus, we could conclude that the central memory size is important in increasing throughput, but not sufficiently enough to match the input rate ; it would be uncovered in later analysis that increasing central memory size relieves the CM queue but increases the sizes of Disk and Cpu queues. But this could happen only at very high mean arrival rate of jobs. For mean arrival rate of jobs with mean inter arrival times of 15 secs. or 14 secs., the throughput matches with the input rate at 250K central memory size, and the system is balanced.

5.8.3. Throughput Vs Mean Disk access time.

Referring to appendix A for simulated system output, it is noted that the I/O activities of the job load simulated amounts to 35 % . Hence, it is expected that increasing mean disk access time would improve throughput. Referring to figure 5-12, the following could be inferred:

(1) The throughput increases very rapidly from seek time of 40 ms. to 30 ms.

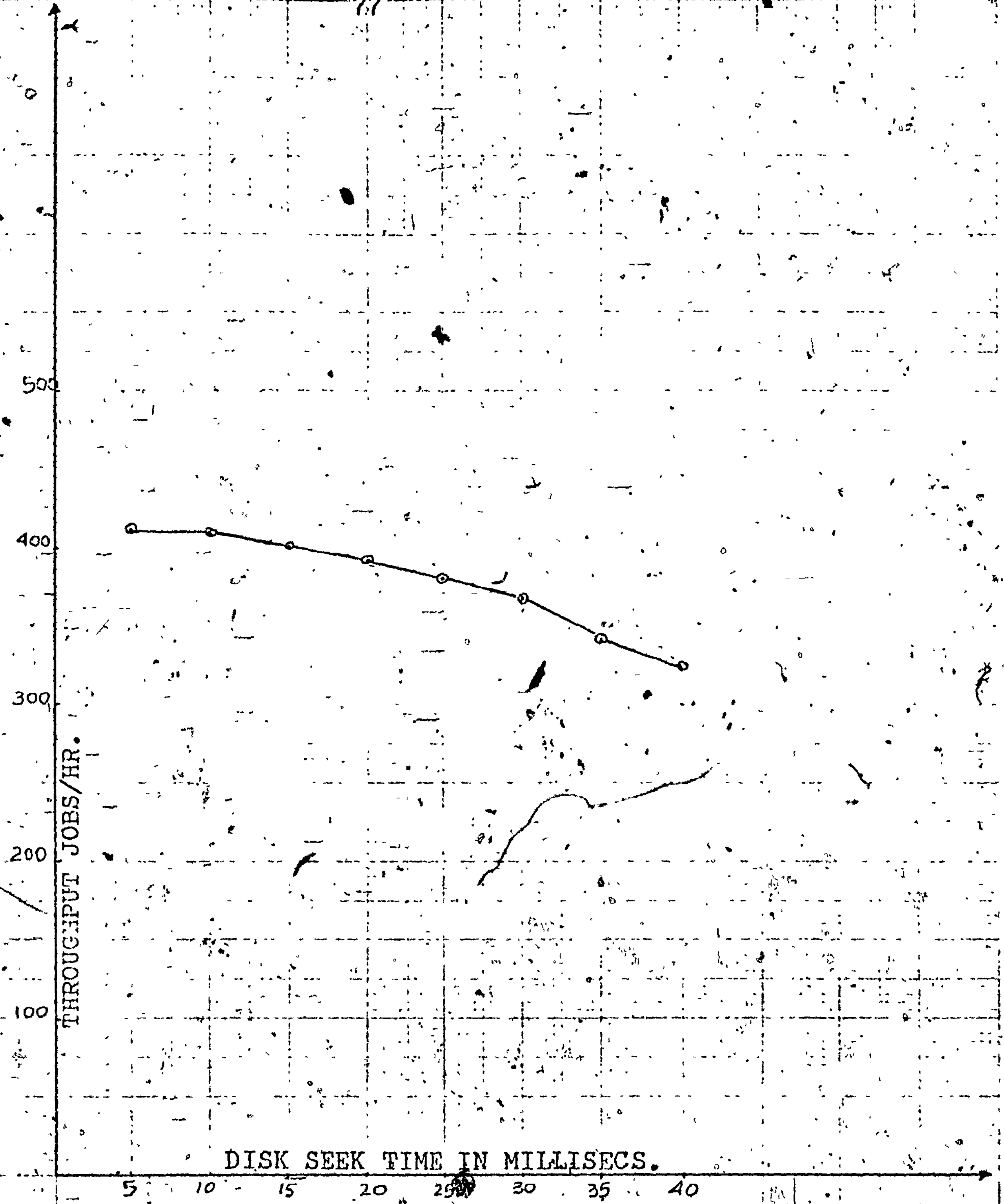


Figure 5-12. Throughput Vs. Average Disk seektime.

(2) The throughput increases very slowly from seek time variations of 40ms. to 5ms.

(3) With mean seek time of 17.5ms, we achieve a throughput of almost 400 jobs/hour.

The fast rise of throughput within a short decrease of average seek time may be accou nted for short I/O bound jobs locked in Disk queue, which are cleared from Disk queue and serviced due to shorter average Disk seek time. The long I/O bound jobs in Disk queue take relatively short time due to improved Disk access time but the time difference is not large enough to augument throughput by large amount. This could be a possible explanation of slow rise in throughput between 30ms. to 5ms. range of average seek time. In this experimentation the mean Disk seek time is varied. To reduce the effect of rotational time on measurement , its average value is reduced to 10ms. for Disk seek time 20ms. The Word transfer time is taken as 125000 characters/sec.

5.8.4. Effects of Central Memory Size and Mean Disk Access time on System.

The overall effect on system of mean Disk seek time variation is summarised in Table 5-2 and the effect of central memory size is provided in Appendix A as simulated results.

From table 5-2 we could conclude that the mean waiting time in Disk queue is reduced with the reduction of mean Disk seek time but it is intersting to note that the

TABLE 5-2. Variation of queue parameters with Disk seek time

Disk Seek time	Mean Wait Sec.			Max. Wait			Mean Q-length			Max. Q-legth.		
	CM	DK	CPU	CM	DK	CPU	CM	DK	CPU	CM	DK	CPU
5ms	150	1	3	4302	9	46	16		8	44	15	17
10ms	172	1	3	4414	13	43	21	1	8	54	16	18
15ms	177	1	3	5086	15	48	30	1	8	70	18	18
20ms	198	2	3	4244	18	44	44	2	7	89	18	16
25ms	199	2	3	4162	21	46	54	2	8	113	19	19
30ms	188	3	4	5211	24	44	70	3	8	144	19	19

* Job input rate = 8 jobs/min.

mean wait in central memory queue is also reduced. The system is very sensitive to disk seek time. A faster access brings the throughput consideraly higher than what we could achieve by increasing the central memory size. A faster disk access would improve both throughput and response time.

5.8.5. System throughput with time-slicing.

This is the last phase of work with the model.which had to be changed to incorporate Cpu scheduling.

Figure 5-19 shows the flowchart of the model coded in Simscript. The major area of modification was the introduction of a new module / Cpu scheduler, called CPUSR. The main function of CPUSR is to give control of CPU to a job only for a fixed amount of time, and rotate the Cpu among various jobs waiting in central memory. The CPUSR is scheduled by endogenous event PHAS2. The CPUSR may schedule endogenous events PHAS2, PHAS4, or itself.

If Cpu time distributed between I/O has elapsed, it will schedule endogenous event PHAS3. When all I/O activities are over except final output, it will schedule PHAS2. It schedules itself when it assigns Cpu to a waiting job; intuitively this should increase the turn around time of I/O bound jobs. Figure 5-13 shows the relationship

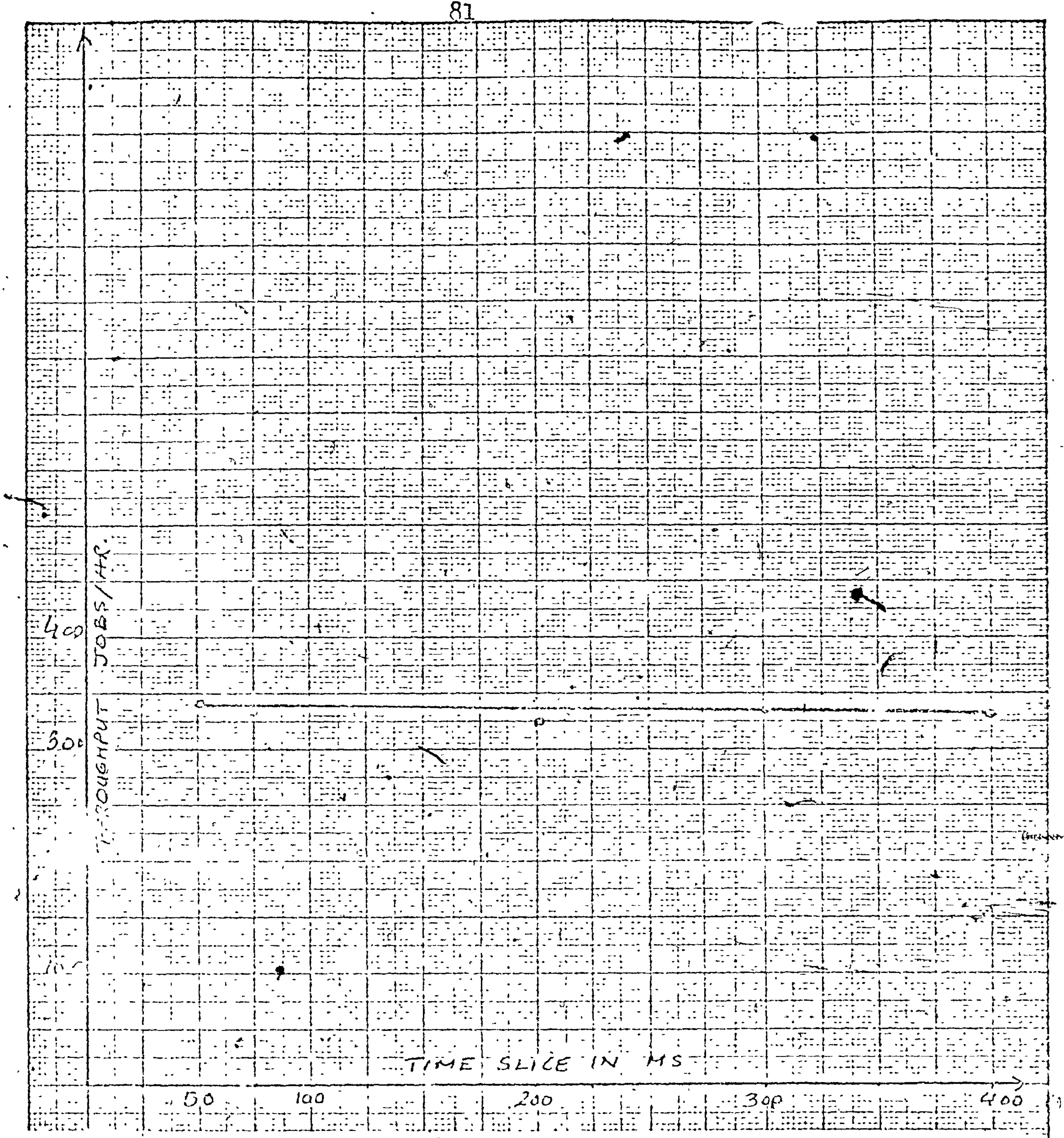


Figure 5-1 3. Throughput Vs Time Slice.

10 MM/CM

between throughput and time-slice. The time-slice is varied from 50 ms. to 400 ms., and its effect on throughput is noted. Referring to figure 5-13, it can be concluded that the time-slice has very little effect on the throughput with the simulated job stream and the model. The job stream used for simulation has mean time of 8 secs. between job arrival. Also the job stream is CPU bound for 95% of the simulated time and I/O bound for 60% of the simulated time, and this is possible due to overlap of I/O operation and Computation.

As dicussed under 5.4.3, the CPU is tied to a job only a fraction of total compute time. The compute time of a job is sliced with I/O activities during which the CPU is released. This is the reason why CPU scheduling did not increase the throughput in figure 5-13. In chapter 4 , it is mentioned how CPU time-slice affect throughput as found by Sherman and et. al. (S2).

5.8.6. Mean Wait Time Vs. Central Memory Size.

Mean wait time of a job is an important parameter in system estimation. Item 5.7 illustrate the significance of this parameter. Figure 5-14 and 5-15 show the effect of central memory size variation on this parameter; it is apparent in both graphs that the relationship is highly nonlinear. The graphs 5-14 and 5-15 show the following:

As the central memory size is increased from 100K onwards with an increment of 50K slices, it is evident that between 100k and 150K, figure 5-15 exhibit a maximum which occurs at

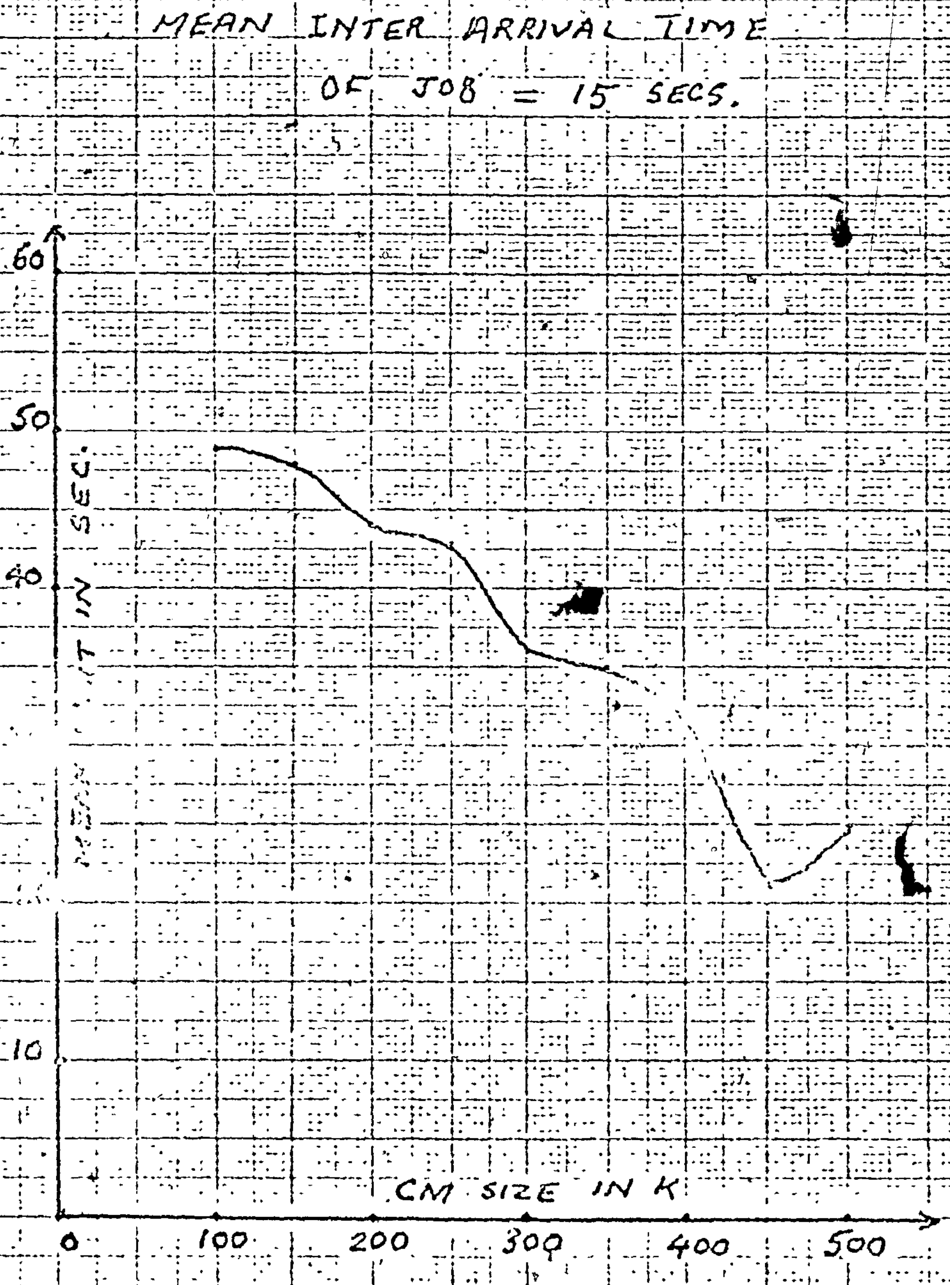


Figure 5- 14 Mean Wait Vs CM Size.

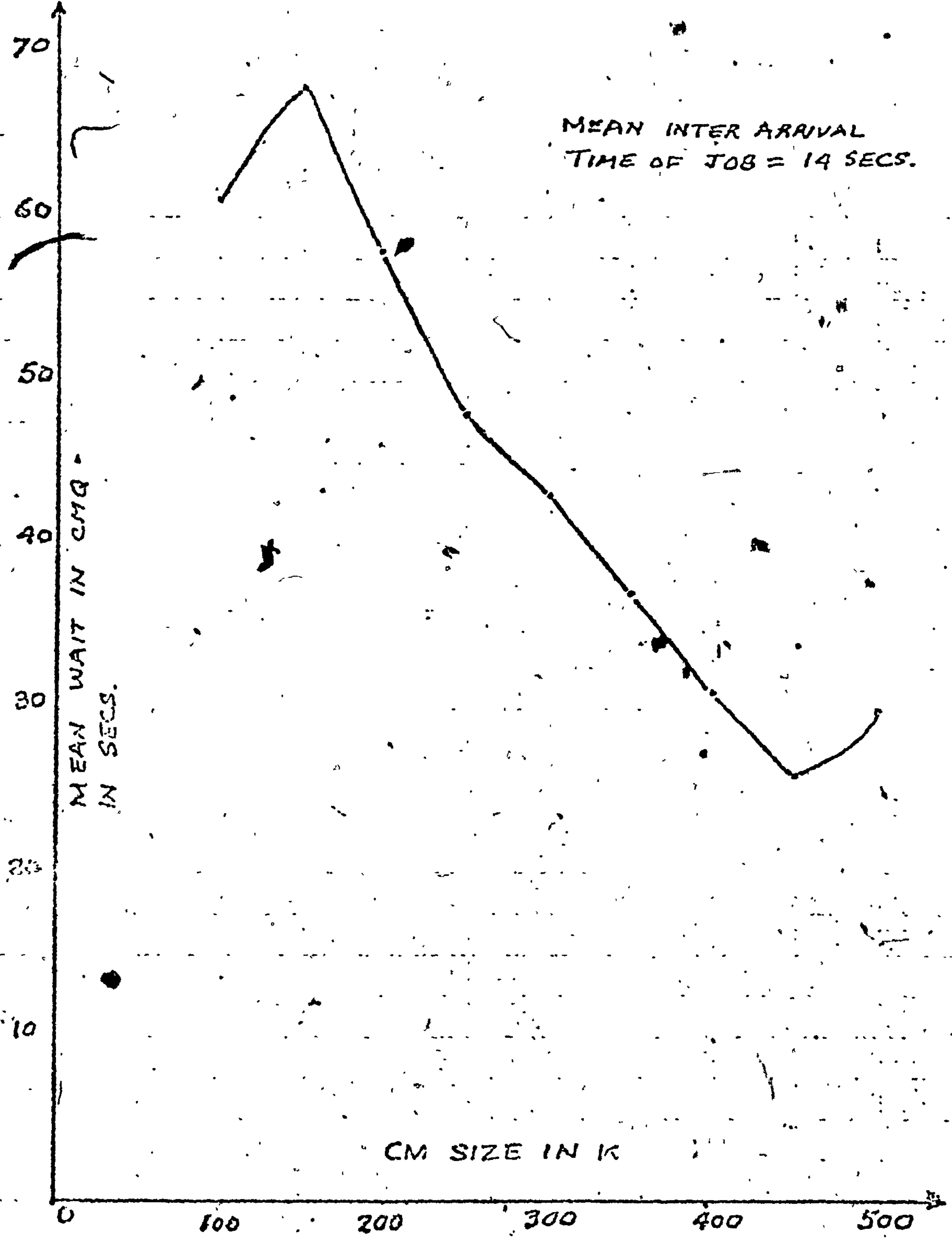


Figure 5- 15 Mean Wait Vs CM Size.

150K and drops to a lower value at 200K. Figure 5-14 and 5-15 demonastrate that, on the whole, the mean waiting time decreases with increasing CM size.

The relationship exhibits local maximum and minimum at different values of central memory. It is also apparent that the mean waiting time depends on the job stream. Figure 5-14 is for a job stream of thousand jobs with mean time between job arrival of 15 secs. and figure 5-15 is for a different job stream with mean time between job arrival of 14 secs. The local maximum and minimum values of mean waiting time with variation of central memory size could be explained on the light of the following hypothesis:

Referring to figures 5-16 and 5-17 which show how the maximum queue lengths of different system queues, eg., central memory, disk and cpu are changing with increasing central memory size. It is shown that the central memory queue size is decreasing with increasing central memory size. This is, offcourse, expected. But this does not predict that the mean waiting time will also decrease since the mean waiting time depends on two factors:

(a) The length of the queue.

(b) The time the queue length is maintained.

A particular job waiting in central memory would have an adverse effect on mean waiting time. A job could be held up in central memory queue lacking available memory. Although the total central memory size is increased, the job occupying central memory

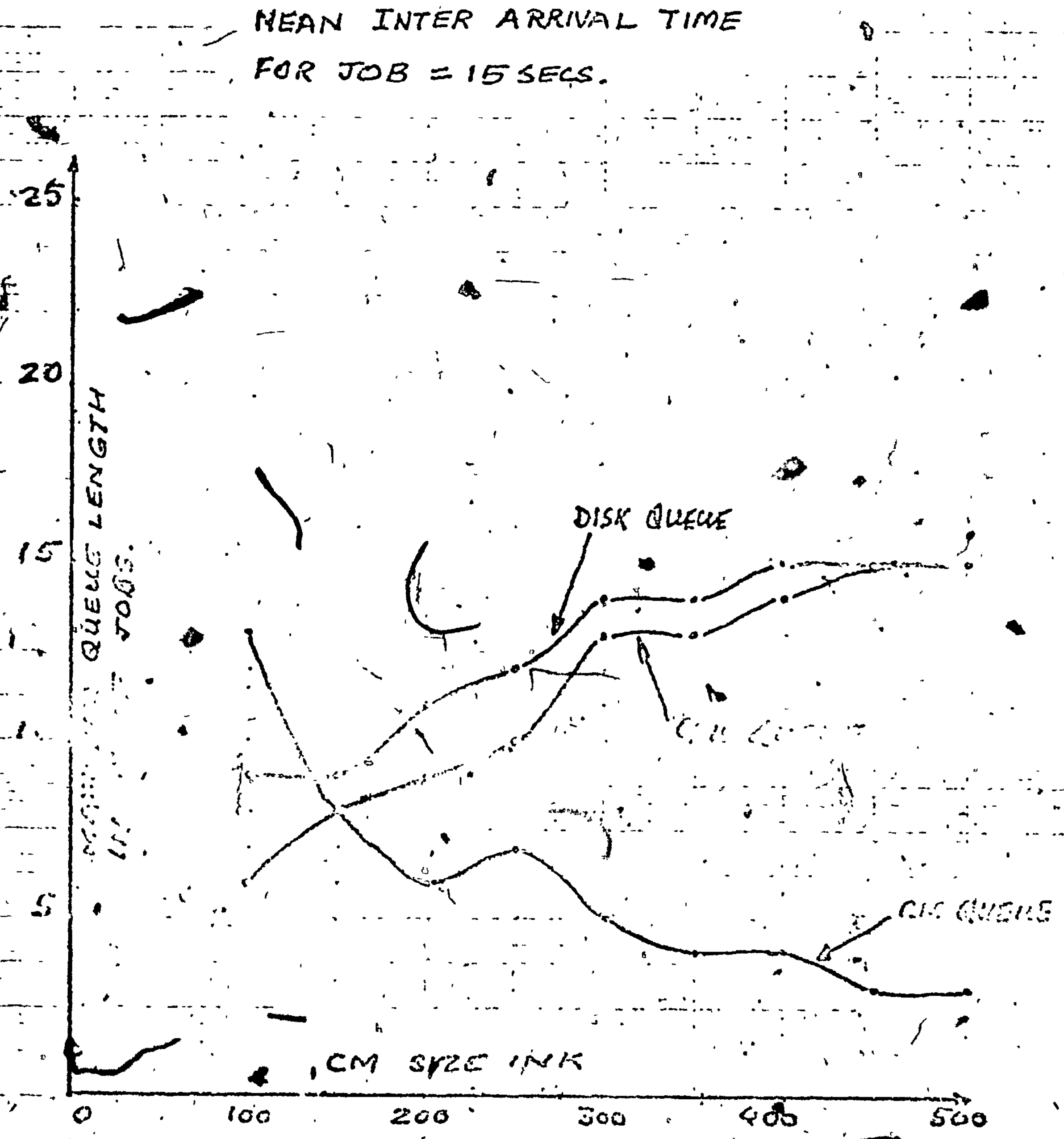


Figure 5-16 . Max. Queue Length Vs CM Size.

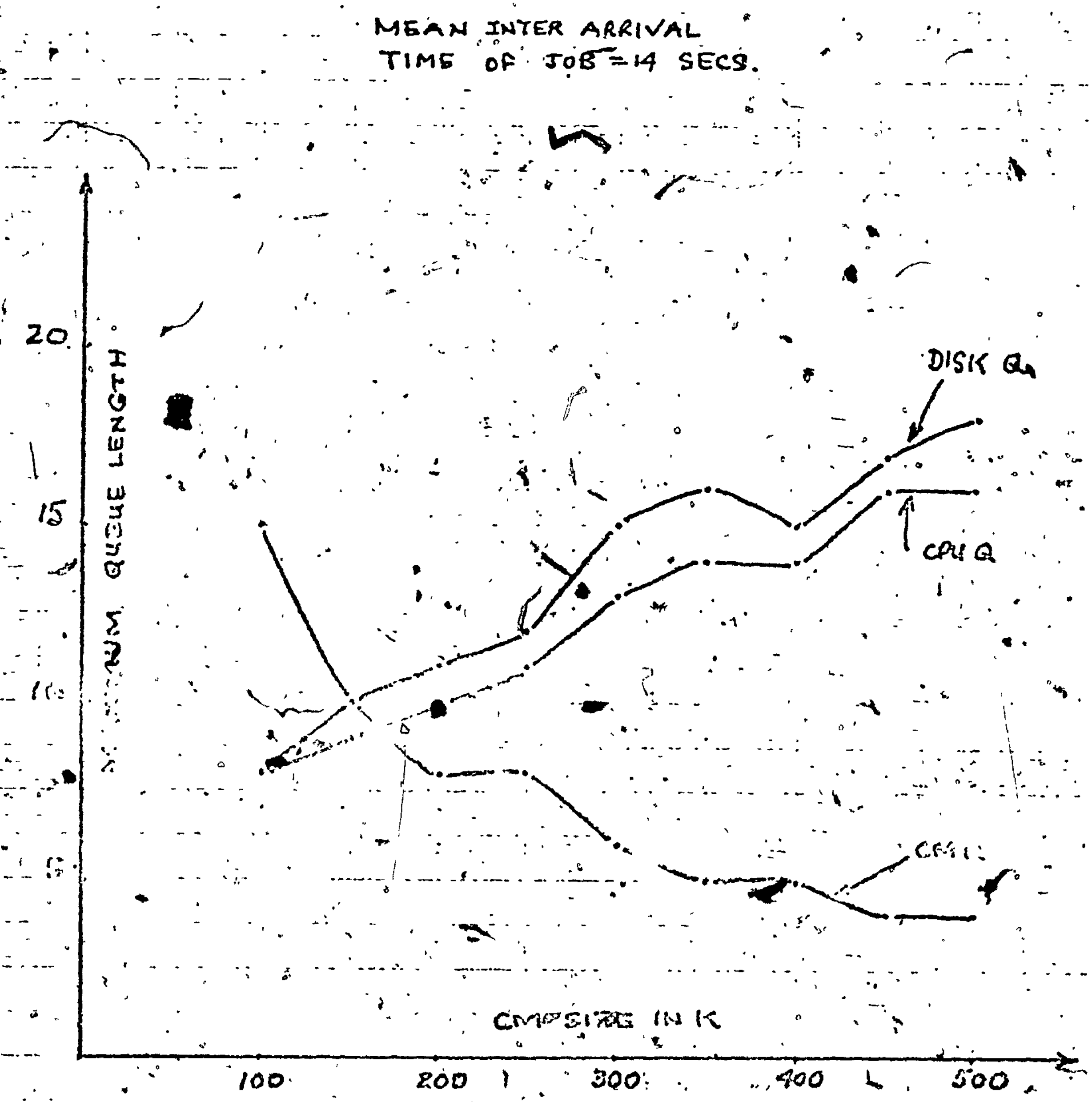


Figure 5-17, Max. Queue Length Vs CM Size.

could be be held there until other resources of the system become available with the result jobs in CM queue would be waiting for longer period of time. Thus it could be deduced that Disk queue and Cpu queue would affect the mean waiting time in central memory queue. This is supported in in the figures 5-16 and 5-17 which show how Disk queue and Cpu queue sizes are increasing with increasing central memory size. Thus central memory alone can not decrease the mean waiting time.

When the system is working against a backlog of jobs in different queues, increasing central memory would result in more jobs being scheduled by the system scheduler in central memory. This would undoubtly decrease central memory queue length but the scheduled jobs would require the services of CPU and DISK, consequently , as shown in figures 5-16 and 5-17, the Disk queue and Cpu queue would increase. Although central memory queue length is decreased , Disk and Cpu queues would cause delay for jobs running into completion and thereby increasing mean waiting time.

It is not possible to predict where the local maximum and minimum would occur for all job stream. The relationship holds in general, overall decrease in wait time with increasing CM size, for different job stream, but exact nature of the curve depends to a large extent on nature of jobs. This is supported in figures 5-14 and 5-15.

CHAPTER 6

6.0. Conclusions.

The performance measurement and evaluation problem is complicated by the large number of variables associated with it. The measurement criteria for performance, as used in this work, are throughput, response time and various queue parameters.

A large multiprogramming and time-sharing system, KRONOS, has been investigated.

Analytical and logical models of multiprogramming system have been developed and studied. Towards this end, the approach was in four phases. The first was devoted in building the logical model; two slightly different models have been developed. The second phase was to implement the model using simulation language, Simscript. The third and the fourth phases were to use the model to understand the difficult -to-understand problems of multiprogramming systems.

The interplay of different parameters in systems are extremely complicated; analytical models are hard to build because the relationship between parameters is not a function but results from structural interactions.

In this work the target is two parameters, throughput and mean waiting time. The objective is to reach an optimum state for these parameters, and in order to achieve that the system resources, incoming job stream pattern and finally

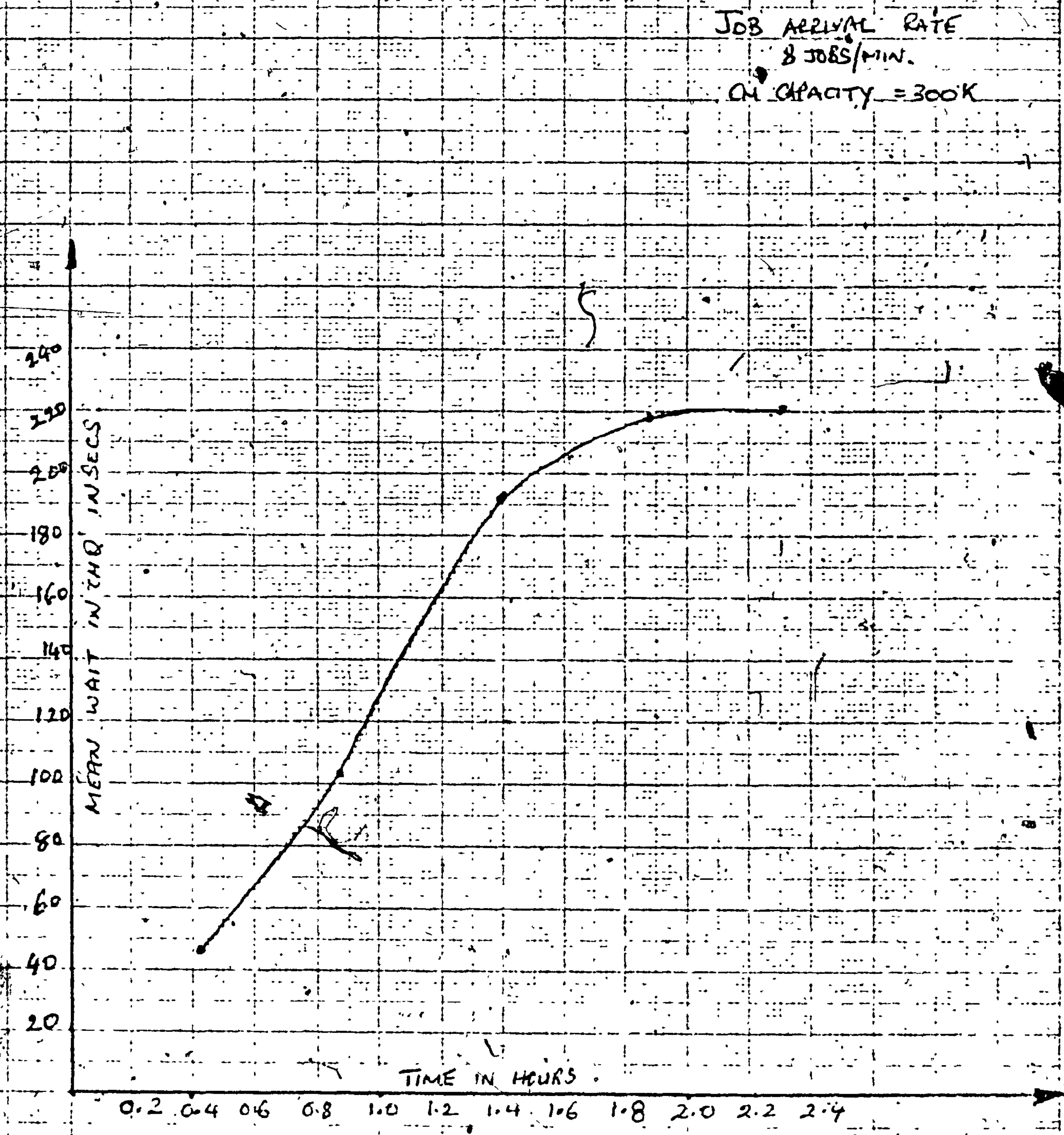


Figure 6-1 Buildup of Mean Wait Time in CMQ.

10 MM/CM

CPU time slice is varied. It was found that there exists a job arrival rate which causes the maximum throughput but Mean Waiting time in Central memory queue deteriorates. Figure 6-1 shows a build up of wait time with progression of time and finally the wait time attains a steady level.

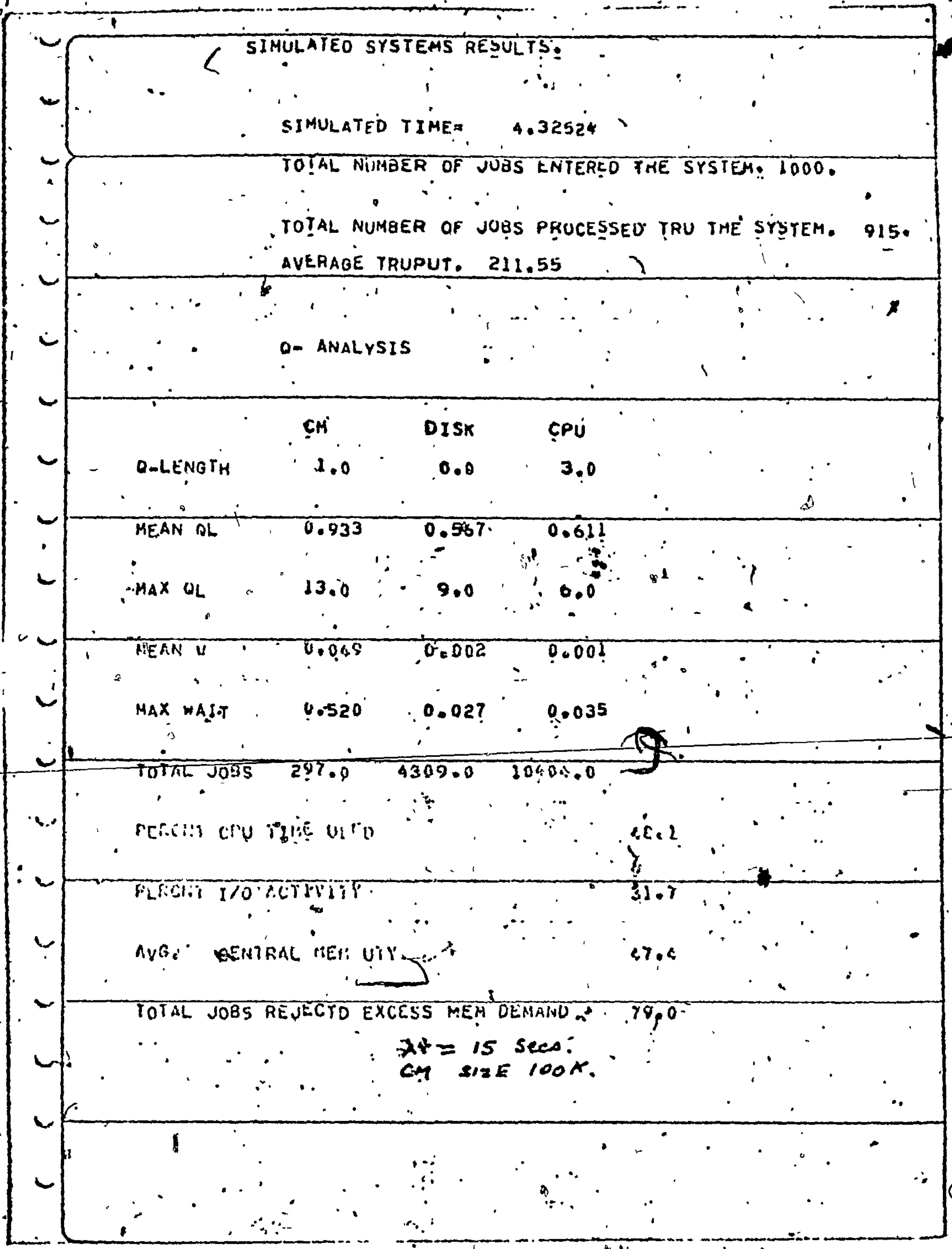
An interesting fact noted in these experimentations was the effect of Disk seek time on central memory queue; a faster seek improves both throughput and wait time. The effect of central memory size after some level, notable 300K, was less prominent than Disk seek time.

One shortcoming of the model is its inability to handle rollout and rollin. It is anticipated that rollout and rollin would affect the Disk queues and it could be a determining factor inthe end. Finally emphasis should be given to the approach to the problem of performance evaluation rather than the specific model and results. The logical model seems to provide the answer towards simplifying the problem a great deal. It also involves the structure of the problem which in the realistic part of the problem into investigation.

In future both the analytical and logical model would be incorporated into one for exact representation of the system. It is anticipated that the present work has generated some questions and thinking on the subject of performance evaluation of a large system, and it has answered a few questions to people who are interested in this area.

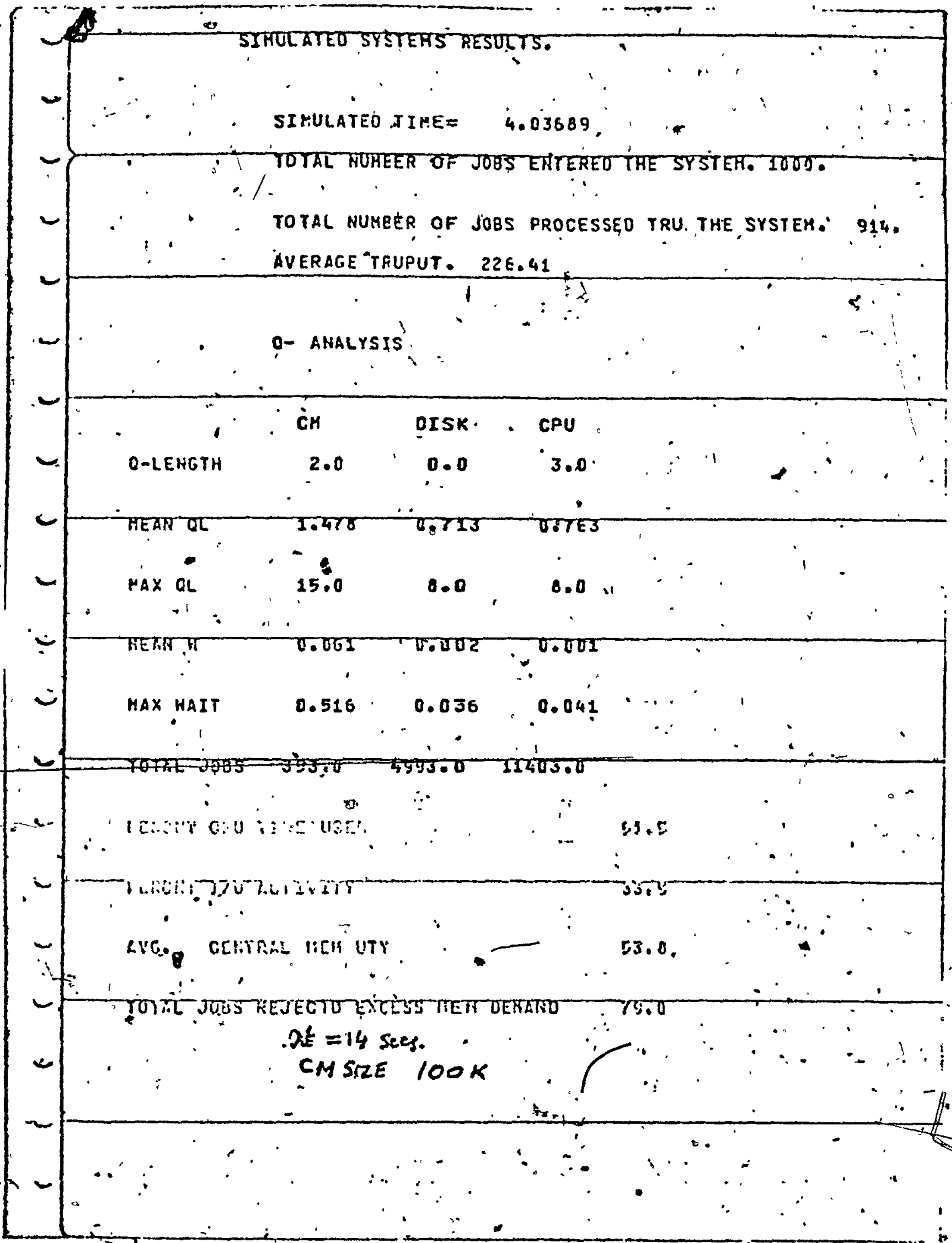
APPENDIX A

APPENDIX A



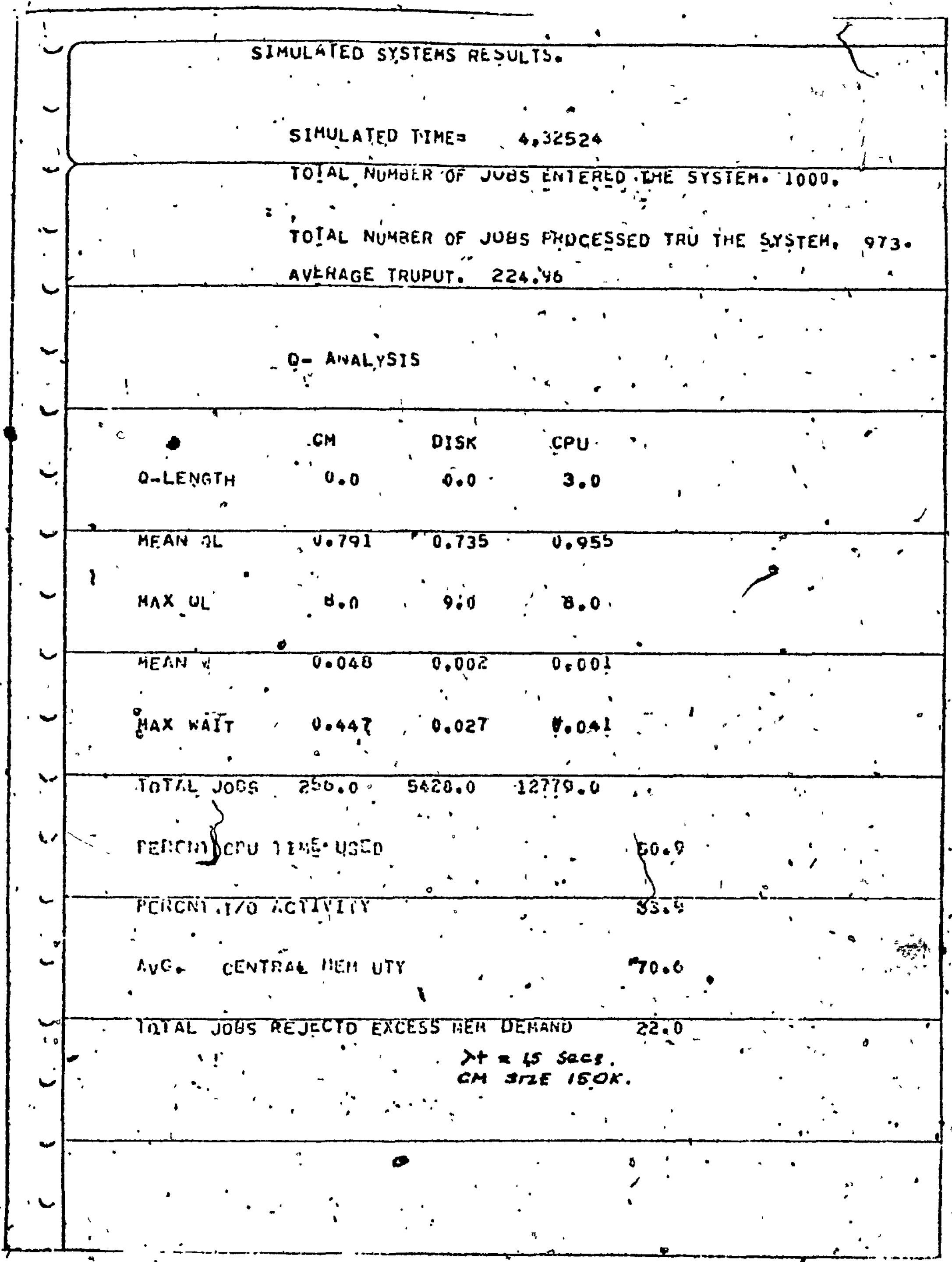
```
SIMULATED SYSTEMS RESULTS.


               SIMULATED TIME=    4.32524

               TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

               TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  915.
               AVERAGE TRUPUT.  211.55


               Q- ANALYSIS


                     CM         DISK       CPU
     Q-LENGTH        1.0        0.0        3.0

     MEAN QL         0.933      0.567      0.611

     MAX QL          13.0       9.0        6.0

     MEAN W          0.069      0.002      0.001

     MAX WAIT        0.520      0.027      0.035

     TOTAL JOBS   297.0     4309.0    10404.0

     PERCNT CPU TIME USED                           48.1

     PERCNT I/O ACTIVITY                            31.7

     AVG. CENTRAL MEM UTY                           47.4

     TOTAL JOBS REJECTD EXCESS MEM DEMAND           79.0
```

λ = 15 Secs.
CM SIZE 100K.



```
        SIMULATED SYSTEMS RESULTS.

             SIMULATED TIME=    4.03689

             TOTAL NUMEER OF JOBS ENTERED THE SYSTEM. 1000.

             TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  914.
             AVERAGE TRUPUT.  226.41


             Q- ANALYSIS


                         CM         DISK     CPU
   Q-LENGTH              2.0        0.0      3.0

   MEAN QL               1.478      0.713    0.763

   MAX QL                15.0       8.0      8.0

   MEAN W                0.061      0.002    0.001

   MAX WAIT              0.516      0.036    0.041

   TOTAL JOBS            393.0      4993.0   11403.0

   PERCENT CPU TIME USED                         53.5

   PERCENT I/O ACTIVITY                          33.9

   AVG. CENTRAL MEM UTY                          53.8.

   TOTAL JOBS REJECTD EXCESS MEM DEMAND          79.0
```

Δt =14 secs.
CM SIZE 100K

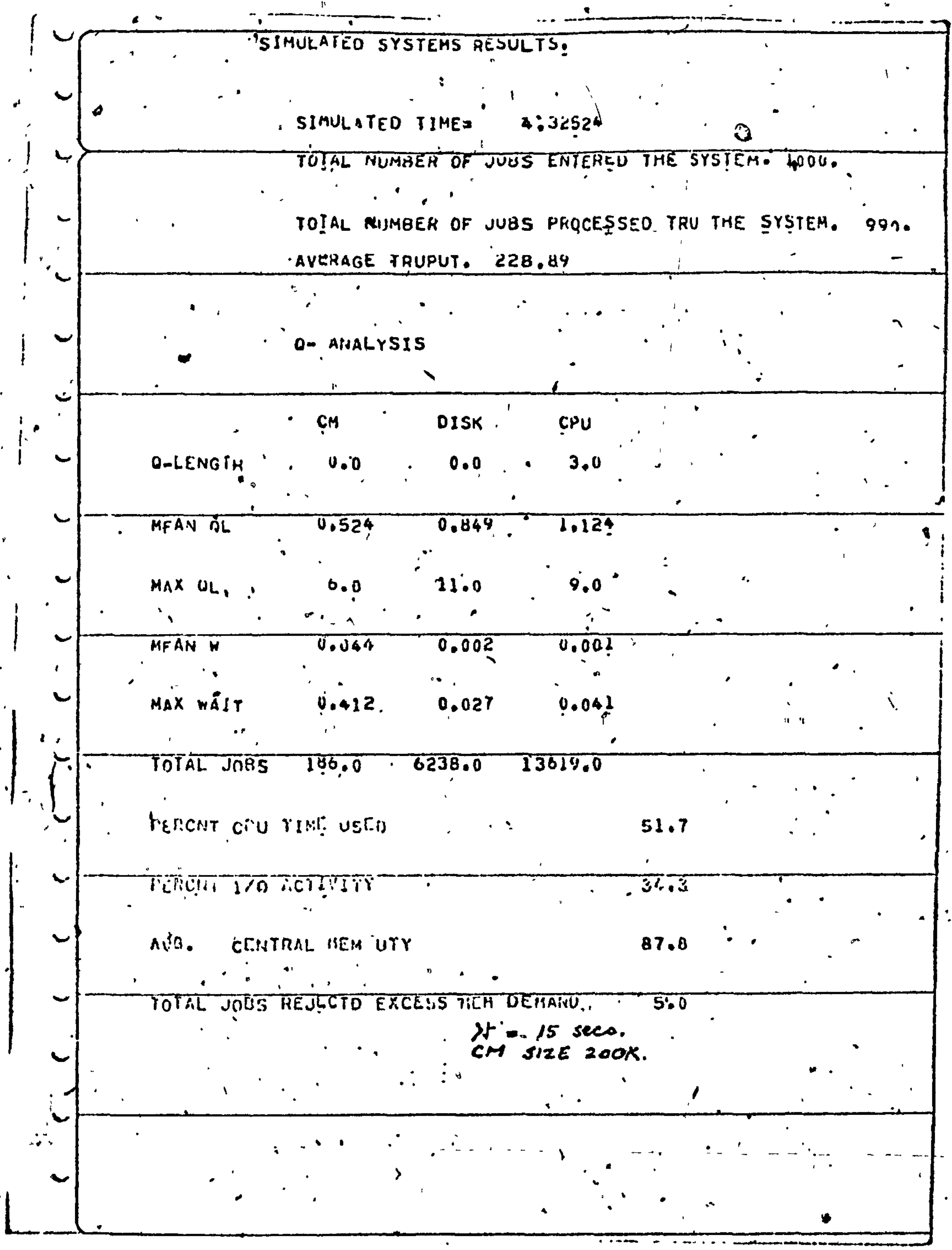


```
                SIMULATED SYSTEMS RESULTS.


                   SIMULATED TIME=     4,32524

                   TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

                   TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM,  973.

                   AVERAGE TRUPUT.  224.96



                   Q- ANALYSIS


                         CM          DISK        CPU

     Q-LENGTH            0.0          0.0         3.0

     MEAN QL            0.791        0.735       0.955

     MAX QL              8.0          9.0         8.0

     MEAN W             0.048        0.002       0.001

     MAX WAIT           0.447        0.027       0.041

     TOTAL JOBS        256.0       5428.0     12779.0

     PERCNT CPU TIME USED                                 60.9

     PERCNT I/O ACTIVITY                                  33.9

     AVG.   CENTRAL MEM UTY                               70.6

     TOTAL JOBS REJECTD EXCESS MEM DEMAND                 22.0
```

λt = 15 Secs.
CM SIZE 150K.

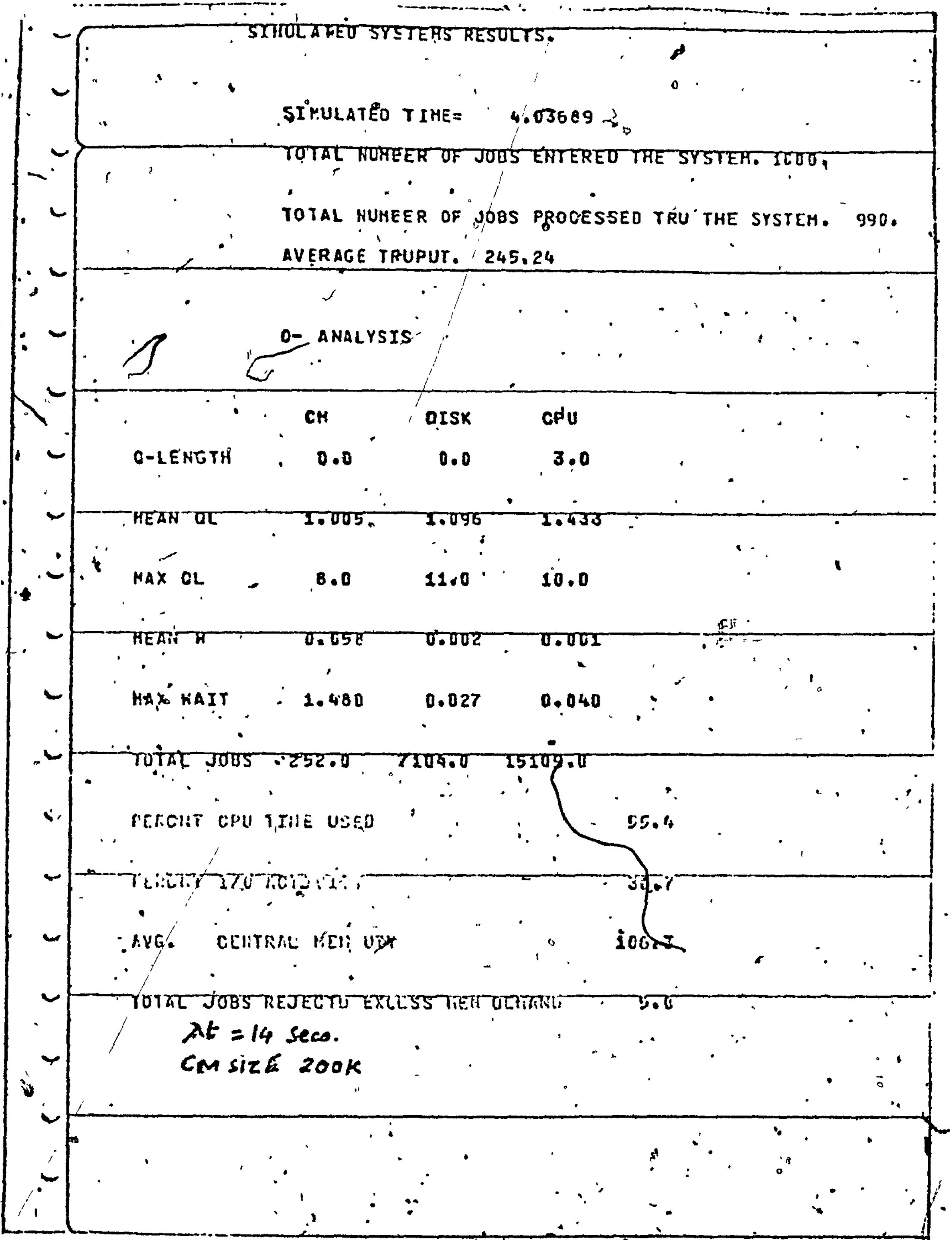
```
SIMULATED SYSTEMS RESULTS.

SIMULATED TIME=     4.03689
TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  973.
AVERAGE TRUPUT.  241.03

Q- ANALYSIS

                 CM          DISK       CPU
Q-LENGTH         0.0          0.0        2.0

MEAN QL         1.526        0.959      1.163

MAX QL          10.0         10.0        9.0

MEAN W          [illegible]  0.002      0.002

MAX WAIT        0.637        0.036      0.041

TOTAL JOBS      [illegible]  [illegible] [illegible]

PERCNT CPU TIM USED                        54.0

PERCNT I/O ACTIVITY                        36.3

AVG.  CENTRAL MEM UTY                      83.9

TOTAL JOBS REJECTD EXCESS MEM DEMAND       22.0
```

λt=14 Secs.
CM SIZE 150K.



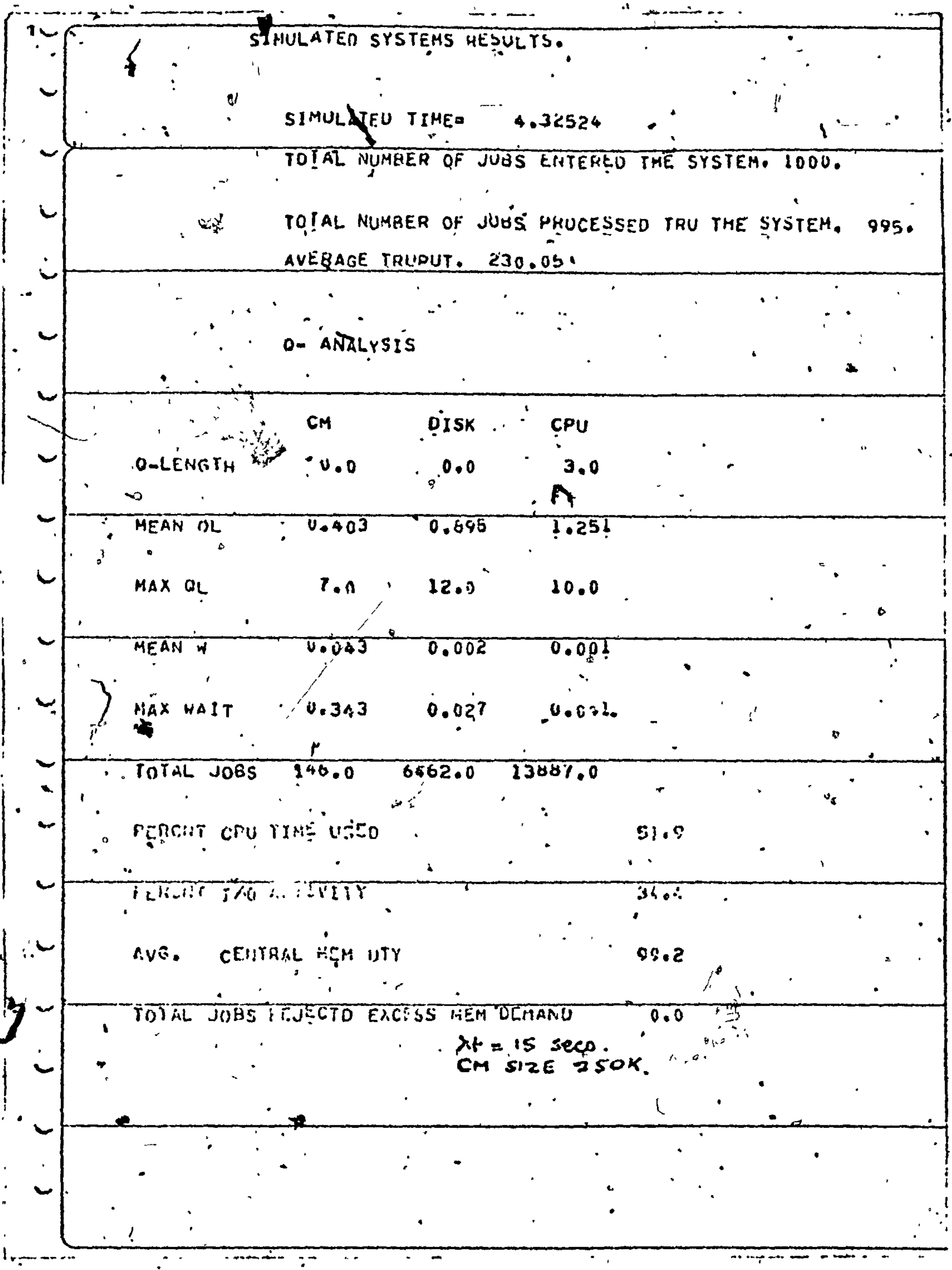
```
          SIMULATED SYSTEMS RESULTS.

             SIMULATED TIME=     4.32524
             TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

             TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  990.
             AVERAGE TRUPUT.  228.89

             Q- ANALYSIS

                         CM           DISK         CPU
   Q-LENGTH              0.0          0.0          3.0

   MEAN QL               0.524        0.849        1.124

   MAX QL,               6.0          11.0         9.0

   MEAN W                0.044        0.002        0.001

   MAX WAIT              0.412        0.027        0.041

   TOTAL JOBS         186.0       6238.0     13619.0

   PERCNT CPU TIME USED                                 51.7

   PERCNT I/O ACTIVITY                                  34.3

   AVG.   CENTRAL MEM UTY                               87.8

   TOTAL JOBS REJECTD EXCESS MEM DEMAND.                 5.0
```

λ = .15 secs.
CM SIZE 200K.



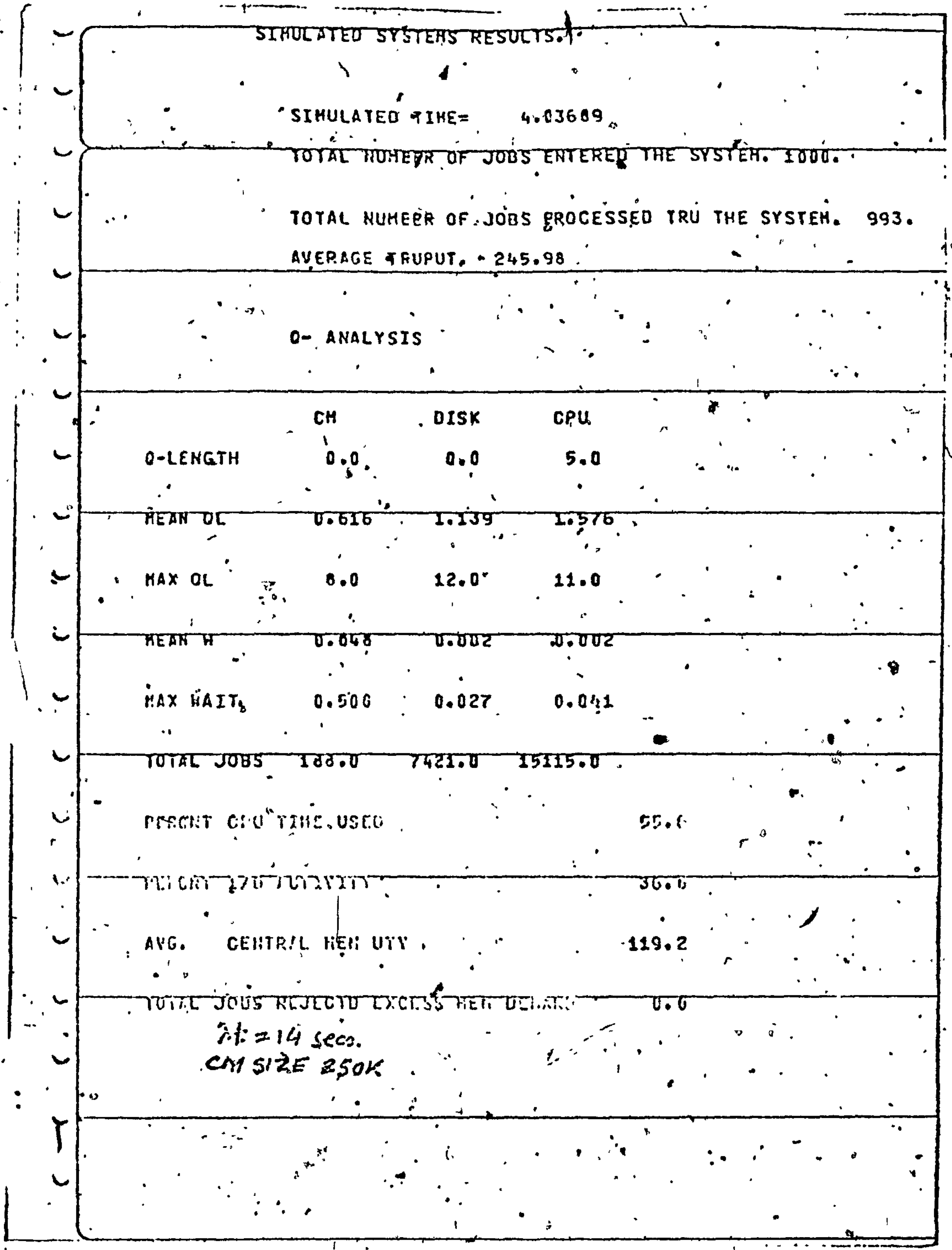
```
SIMULATED SYSTEMS RESULTS.

        SIMULATED TIME=     4.03689

        TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

        TOTAL NUMEER OF JOBS PROCESSED TRU THE SYSTEM.  990.

        AVERAGE TRUPUT.  245.24

        Q- ANALYSIS

                      CM          DISK         CPU

 Q-LENGTH             0.0          0.0          3.0

 MEAN QL              1.005        1.096        1.433

 MAX QL               8.0         11.0         10.0

 MEAN W               0.058        0.002        0.001

 MAX WAIT             1.480        0.027        0.040

 TOTAL JOBS         252.0       7104.0      15109.0

 PERCNT CPU TIME USED                                  55.4

 PERCNT I/O ACTIVITY                                   35.7

 AVG.    CENTRAL MEM UTX                              100.3

 TOTAL JOBS REJECTD EXCESS MEM DEMAND                   5.0
```

λt = 14 Secs.
CM SIZE 200K



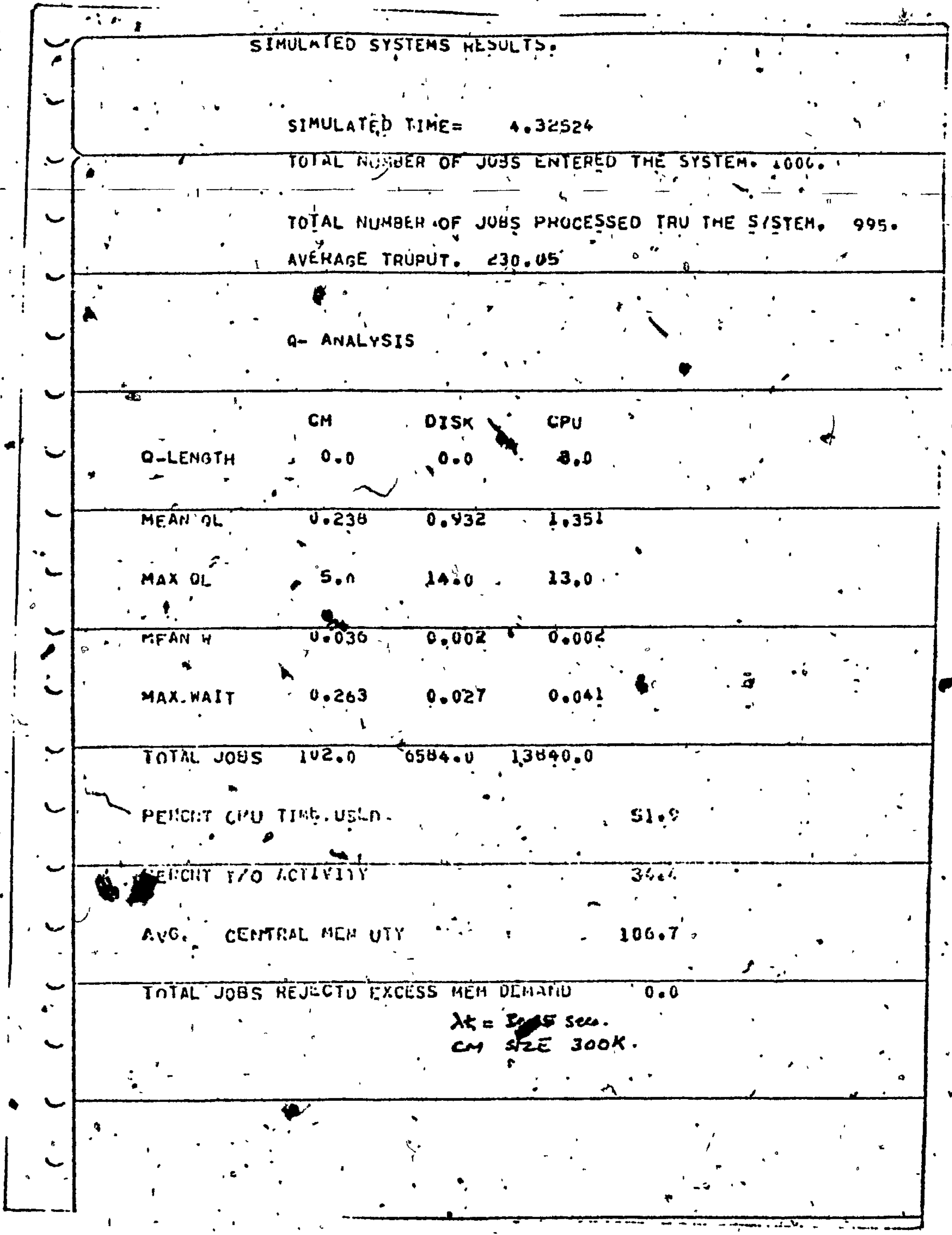
```
SIMULATED SYSTEMS RESULTS.


     SIMULATED TIME=    4.32524
     TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

     TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  995.
     AVERAGE TRUPUT.  230.05


     Q- ANALYSIS

                  CM          DISK        CPU
Q-LENGTH          0.0         0.0         3.0

MEAN QL           0.403       0.695       1.251

MAX QL            7.0         12.0        10.0

MEAN W            0.043       0.002       0.001

MAX WAIT          0.343       0.027       0.041

TOTAL JOBS      146.0      6662.0     13887.0

PERCNT CPU TIME USED                                51.9

PERCNT I/O ACTIVITY                                 34.4

AVG.   CENTRAL MEM UTY                              99.2

TOTAL JOBS REJECTD EXCESS MEM DEMAND                 0.0
```

λt = 15 secs.
CM SIZE 250K.



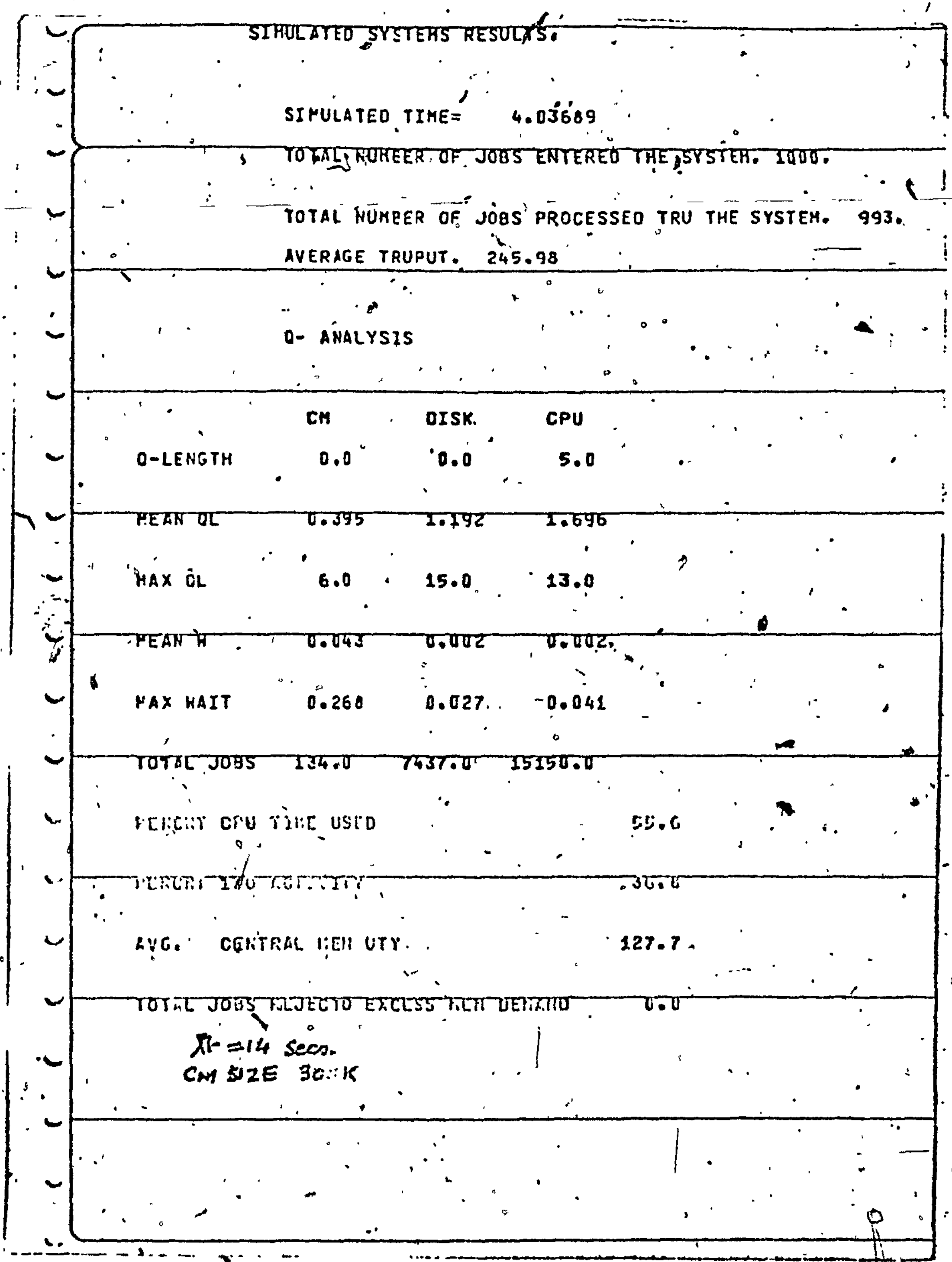
```
          SIMULATED SYSTEMS RESULTS.

             SIMULATED TIME=     4.03689

             TOTAL NUMEBR OF JOBS ENTERED THE SYSTEM. 1000.

             TOTAL NUMEER OF JOBS PROCESSED TRU THE SYSTEM.  993.

             AVERAGE TRUPUT. = 245.98

             Q- ANALYSIS

                         CM         DISK        CPU
     Q-LENGTH           0.0         0.0         5.0

     MEAN QL           0.616       1.139       1.576

     MAX QL             8.0        12.0        11.0

     MEAN W            0.048       0.002       0.002

     MAX WAIT          0.506       0.027       0.041

     TOTAL JOBS   188.0      7421.0     15115.0

     PERCNT CPU TIME USED                           55.6

     PERCNT I/O ACTIVITY                            36.0

     AVG.   CENTRAL MEM UTY                        119.2

     TOTAL JOBS REJECTD EXCESS MEM DEMAND           0.0
```

Δt = 14 secs.
CM SIZE 250K



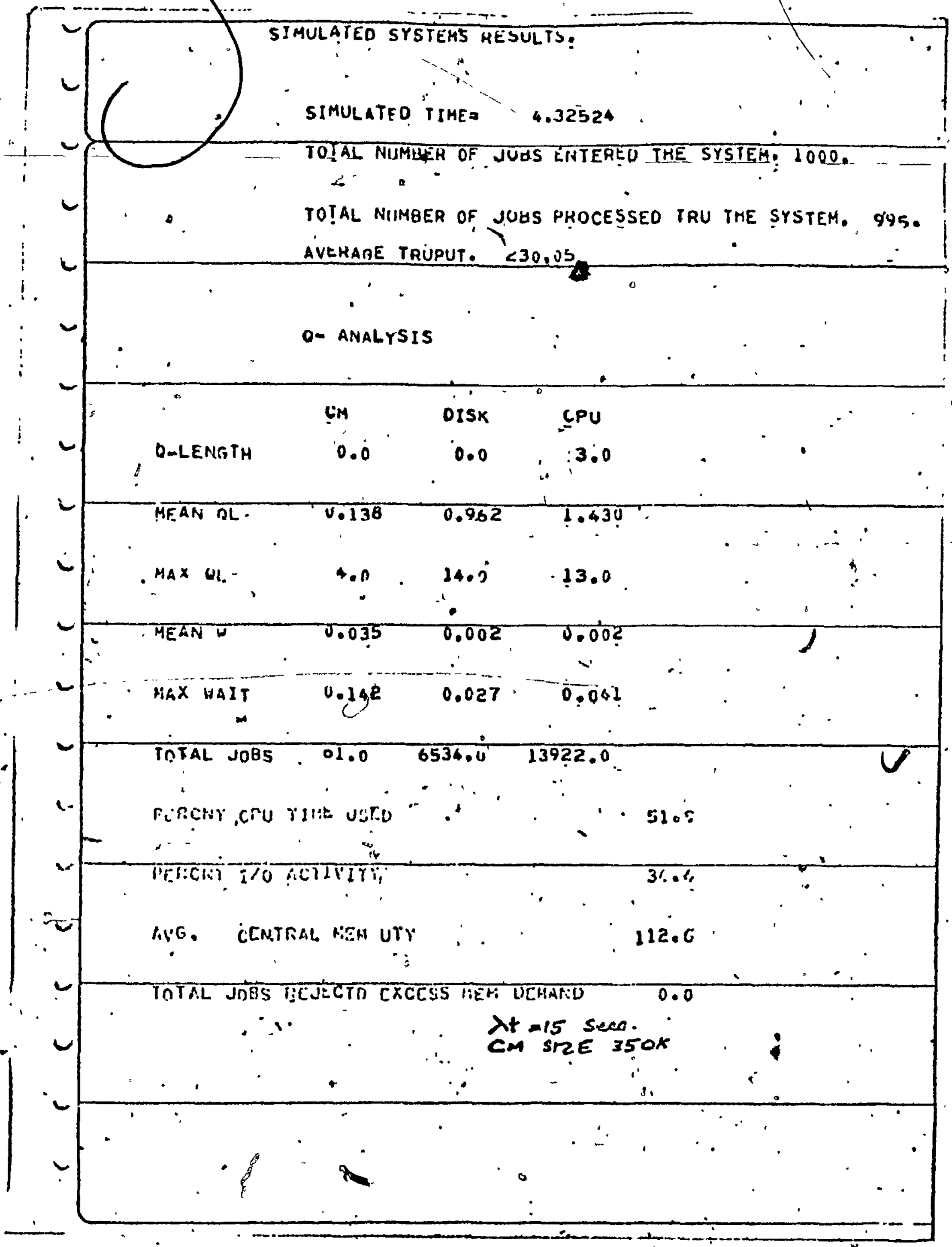
```
               SIMULATED SYSTEMS RESULTS.


                  SIMULATED TIME=     4.32524
                  TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

                  TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  995.
                  AVERAGE TRUPUT.  230.05


                  Q- ANALYSIS


                    CM         DISK        CPU
      Q-LENGTH      0.0         0.0        8.0

      MEAN QL     0.238       0.932      1.351

      MAX QL        5.0        14.0       13.0

      MEAN W      0.036       0.002      0.004

      MAX.WAIT    0.263       0.027      0.041

      TOTAL JOBS  102.0      6584.0    13840.0

      PERCNT CPU TIME.USED.                         51.9

      PERCNT I/O ACTIVITY                           34.4

      AVG.  CENTRAL MEM UTY                        106.7

      TOTAL JOBS REJECTD EXCESS MEM DEMAND          0.0
```

λt = [illegible] sec.
CM SIZE 300K.



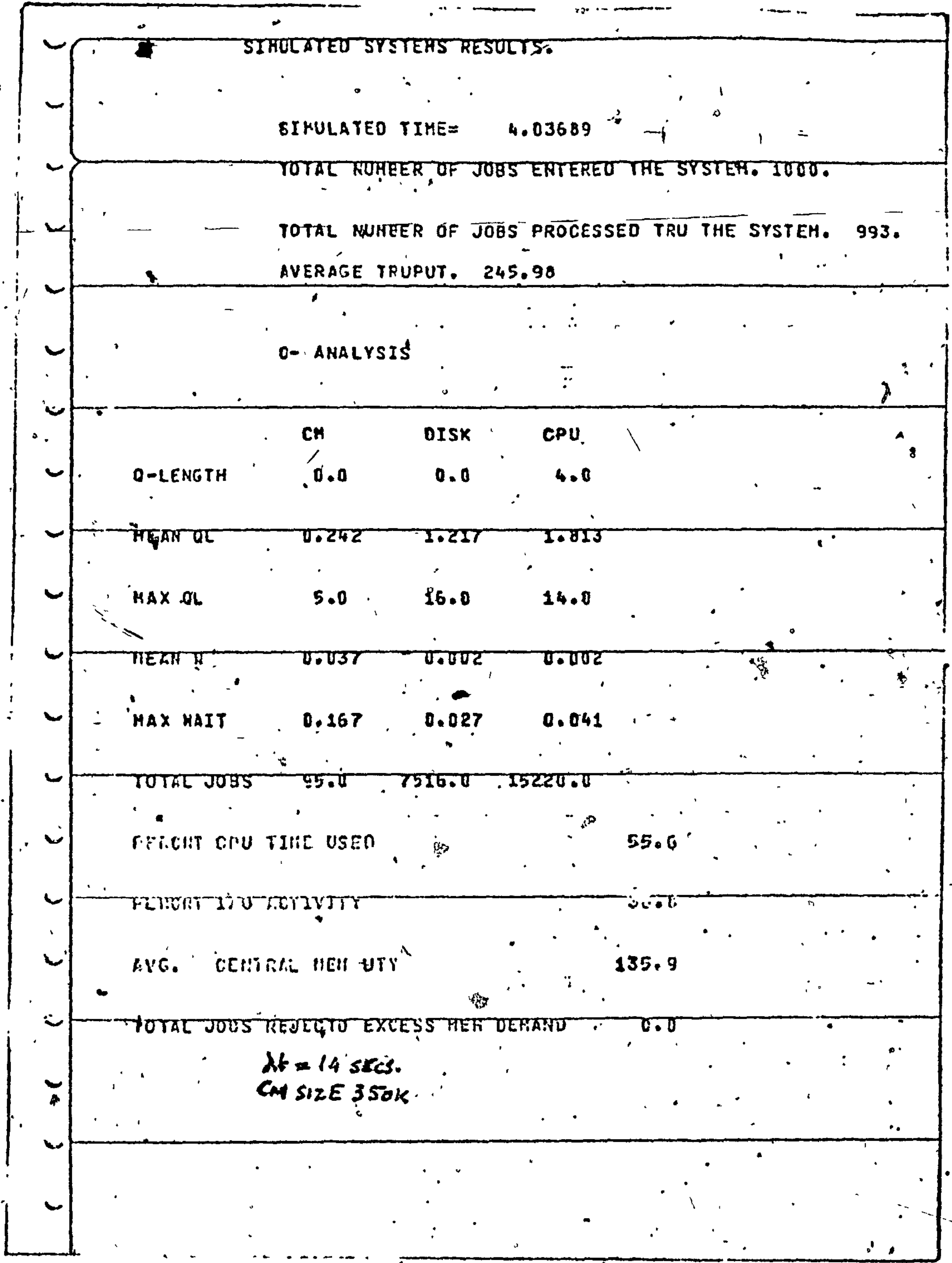
```
SIMULATED SYSTEMS RESULTS.

          SIMULATED TIME=     4.03689
          TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

          TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  993.
          AVERAGE TRUPUT.  245.98

          Q- ANALYSIS

                      CM        DISK.      CPU
  Q-LENGTH           0.0        0.0        5.0

  MEAN QL            0.395      1.192      1.696

  MAX QL             6.0       15.0       13.0

  MEAN W             0.043      0.002      0.002

  MAX WAIT           0.268      0.027      0.041

  TOTAL JOBS       134.0     7437.0    15150.0

  PERCNT CPU TIME USED                           55.6

  PERCNT I/O ACTIVITY                            30.0

  AVG.   CENTRAL MEM UTY                        127.7

  TOTAL JOBS REJECTD EXCESS MEM DEMAND           0.0
```

λ =14 Secs.
CM SIZE 300K



```
SIMULATED SYSTEMS RESULTS.


    SIMULATED TIME=     4.32524

    TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

    TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  995.
    AVERAGE TRUPUT.   230.05


    Q- ANALYSIS


                  CM          DISK        CPU
Q-LENGTH          0.0          0.0        3.0

MEAN QL.          0.138       0.962       1.430

MAX QL.           4.0         14.0       13.0

MEAN W            0.035       0.002       0.002

MAX WAIT          0.142       0.027       0.041

TOTAL JOBS       91.0       6534.0    13922.0

PERCNT CPU TIME USED                           51.9

PERCNT I/O ACTIVITY                            34.4

AVG.   CENTRAL MEM UTY                        112.6

TOTAL JOBS REJECTD EXCESS MEM DEMAND           0.0
```

λt =15 Secs.
CM SIZE 350K



```
        SIMULATED SYSTEMS RESULTS.

             SIMULATED TIME=     4.03689

             TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

             TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  993.

             AVERAGE TRUPUT.  245.98

             Q- ANALYSIS

                        CM          DISK        CPU

 Q-LENGTH               0.0         0.0         4.0

 MEAN QL                0.242       1.217       1.813

 MAX QL                 5.0         16.0        14.0

 MEAN W                 0.037       0.002       0.002

 MAX WAIT               0.167       0.027       0.041

 TOTAL JOBS             95.0        7516.0      15220.0

 PERCNT CPU TIME USED                                 55.6

 PERCNT I/O ACTIVITY                                  [illegible]

 AVG.   CENTRAL MEM UTY                               135.9

 TOTAL JOBS REJECTD EXCESS MEM DEMAND                 0.0
```

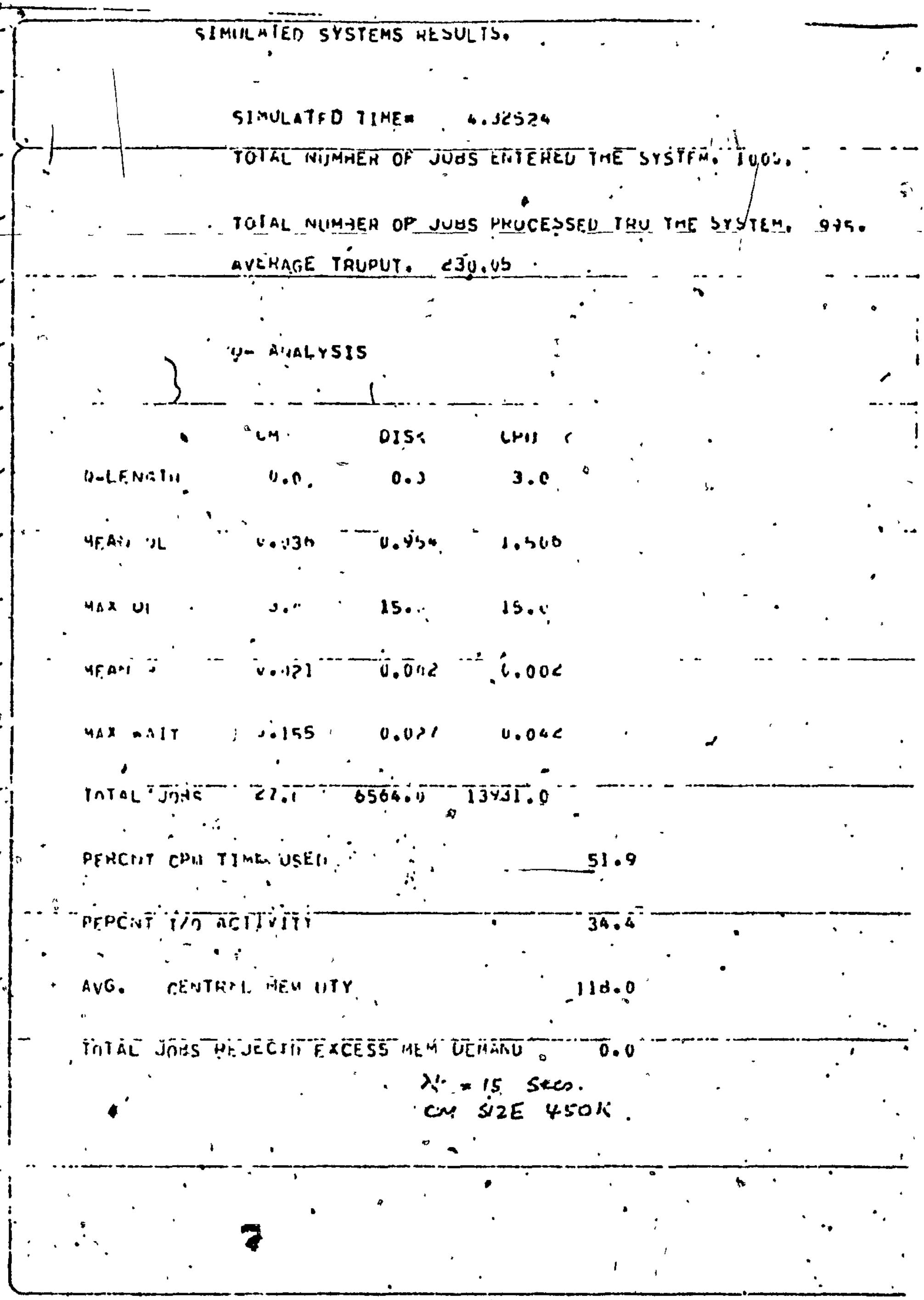
λt = 14 SECS.
CM SIZE 350K

```
          SIMULATED SYSTEMS RESULTS.

             SIMULATED TIME=    4.32524
             TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

             TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  995.
             AVERAGE TRUPUT.  230.05

             Q- ANALYSI

                    VM        DISK       CPU
    Q-LENGTH        0.0       0.0        3.0

    MEAN QL         0.070     0.946      1.471

    MAX QL          4.0       15.0       14.0

    MEAN W          0.032     0.002      0.002

    MAX WAIT        0.159     0.027      0.041

    TOTAL JOBS      34.0      6548.0     13927.0

    PERCNT CPU TIME USED                           51.9

    PERCNT I/O ACTIVITY                            34.4

    AVG.  CENTRAL MEM UTY                          114.9

    TOTAL JOBS REJECTED EXCESS MEM DEMAND          0.0
```

λt = 15 secs.:
CM SIZE 400K

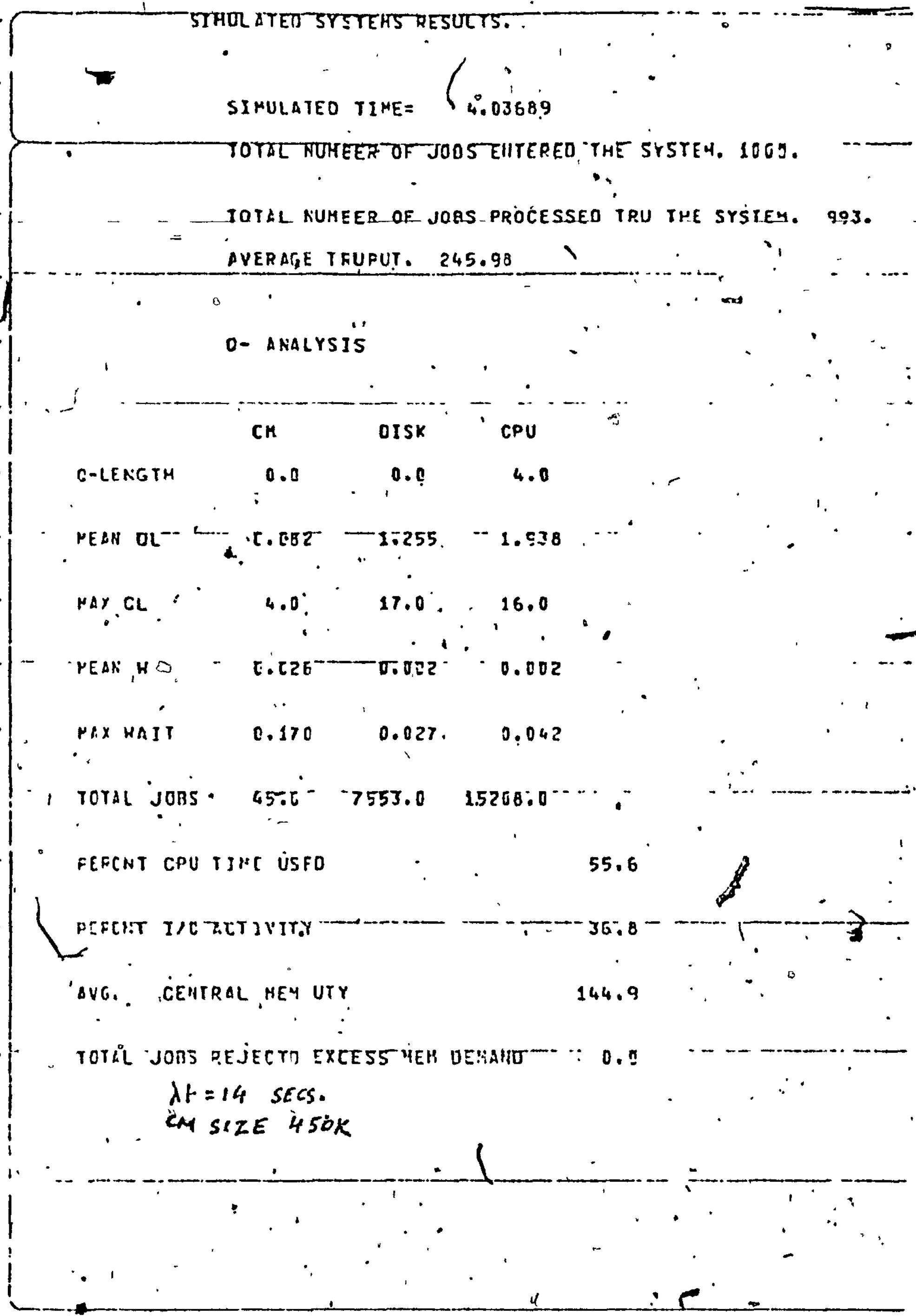
```
        SIMULATED SYSTEMS RESULTS.

          SIMULATED TIME=     4.03649
          TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

          TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  993.
          AVERAGE TRUPUT.  245.98

          C- ANALYSIS

            CM          DISK        CPU
C-LENGTH     0.0         0.0         4.0

MEAN QL      0.136       1.252       1.902

MAX QL       5.0         15.0        14.0

MEAN W       0.031       0.002       0.012

MAX WAIT     0.179       0.027       0.041

TOTAL JOBS   63.0      7590.0     15250.0

PERCNT CPU TIME USED                      55.6

PERCNT I/O ACTIVITY                       36.8

AVG.   CENTRAL MEM UTY                   141.7

TOTAL JOBS REJECTD EXCESS MEM DEMAND       0.0
```

λb = 14 SECS.
CM SIZE 400K



```
SIMULATED SYSTEMS RESULTS.

SIMULATED TIME=    4.32524
TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1005.

TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  995.
AVERAGE TRUPUT.  230.05


Q- ANALYSIS

                CM         DISK        CPU

Q-LENGTH        0.0        0.0         3.0

MEAN QL         0.036      0.954       1.506

MAX QL          3.0        15.0        15.0

MEAN W          0.021      0.002       0.002

MAX WAIT        0.155      0.027       0.042

TOTAL JOBS      27.0       6564.0   13931.0

PERCNT CPU TIME USED                          51.9

PERCNT I/O ACTIVITY                           34.4

AVG.   CENTRAL MEM UTY                       118.0

TOTAL JOBS REJECTD EXCESS MEM DEMAND          0.0
```

λ' = 15 Secs.
CM SIZE 450K.



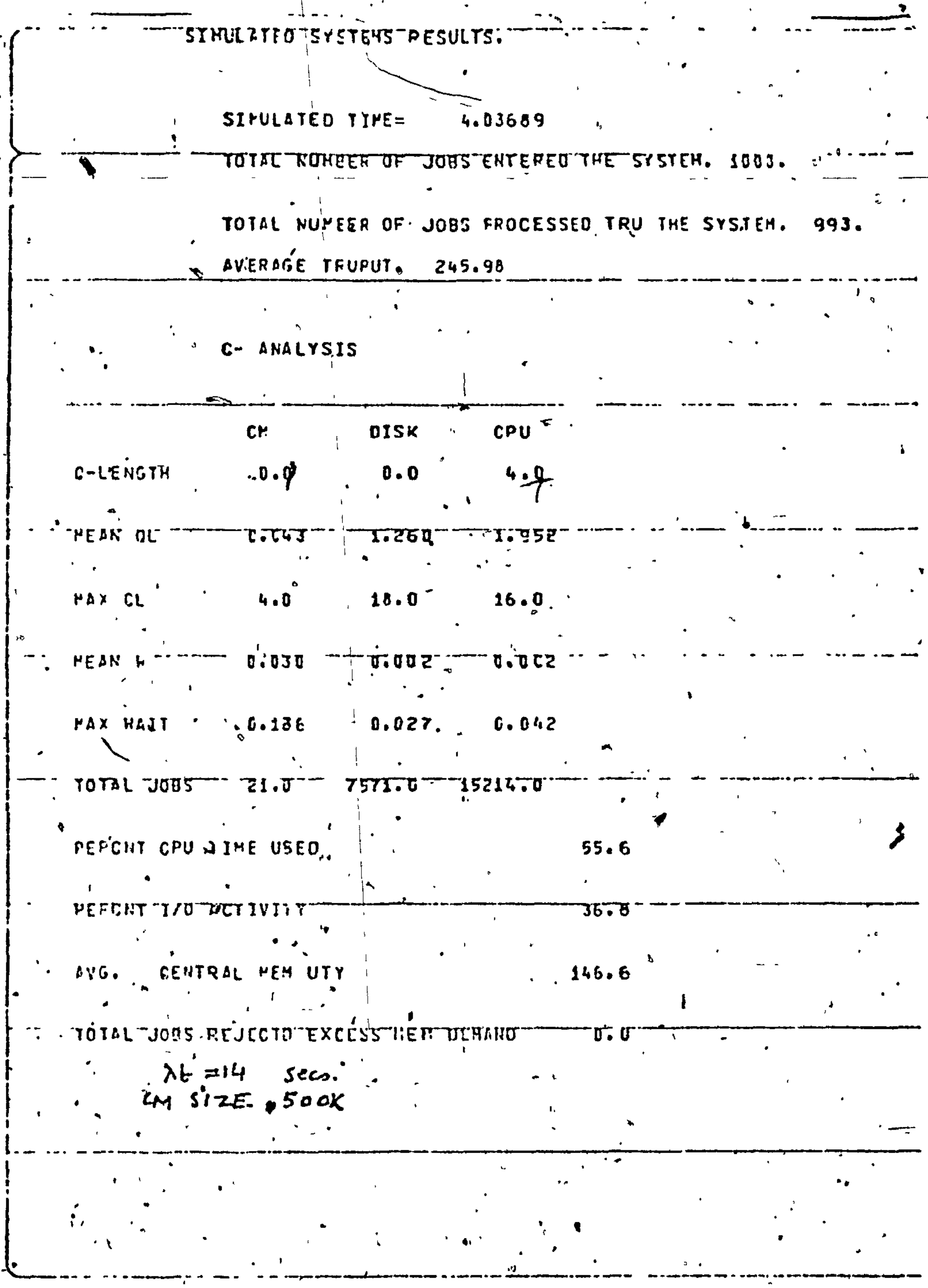
```
SIMULATED SYSTEMS RESULTS.

          SIMULATED TIME=    4.03689
          TOTAL NUMEER OF JOBS ENTERED THE SYSTEM. 1005.

          TOTAL NUMEER OF JOBS PROCESSED TRU THE SYSTEM.  993.
          AVERAGE TRUPUT.  245.98


          Q- ANALYSIS

                 CM         DISK       CPU
   Q-LENGTH      0.0         0.0       4.0

   MEAN QL       0.082       1.255     1.938

   MAX QL        4.0        17.0      16.0

   MEAN WT       0.026       0.022     0.002

   MAX WAIT      0.170       0.027     0.042

   TOTAL JOBS    455.0    7553.0   15208.0

   PERCNT CPU TIME USED                      55.6

   PERCNT I/O ACTIVITY                       36.8

   AVG.  CENTRAL MEM UTY                    144.9

   TOTAL JOBS REJECTD EXCESS MEM DEMAND      0.0
```

λt = 14 SECS.
CM SIZE 450K

λ_2

```
          SIMULATED SYSTEMS RESULTS.


               SIMULATED TIME=    4.325740

               TOTAL  NUMBER OF JOBS ENTERED THE SYSTEM. 1000.

               TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  995.

               AVERAGE TRUPUT.  230.05


               Q- ANALYSIS


                        CM         DISK        CPU

     Q-LENGTH          0.0         0.0         3.0

     MEAN QL          0.018       0.956       1.526

     MAX QL            3.0        15.0        15.0

     MEAN W           0.025       0.002       0.002

     MAX WAIT         0.077       0.027       0.042

     TOTAL JOBS        11.0      6583.0     13937.0

     PERCNT CPU TIME USED                               51.9

     PERCNT I/O ACTIVITY                                34.4

     AVG.   CENTRAL MEM UTY                            119.0

     TOTAL JOBS REJECTD EXCESS MEM DEMAND               0.0
```

λt = 15 secs.
CM SIZE 500K.

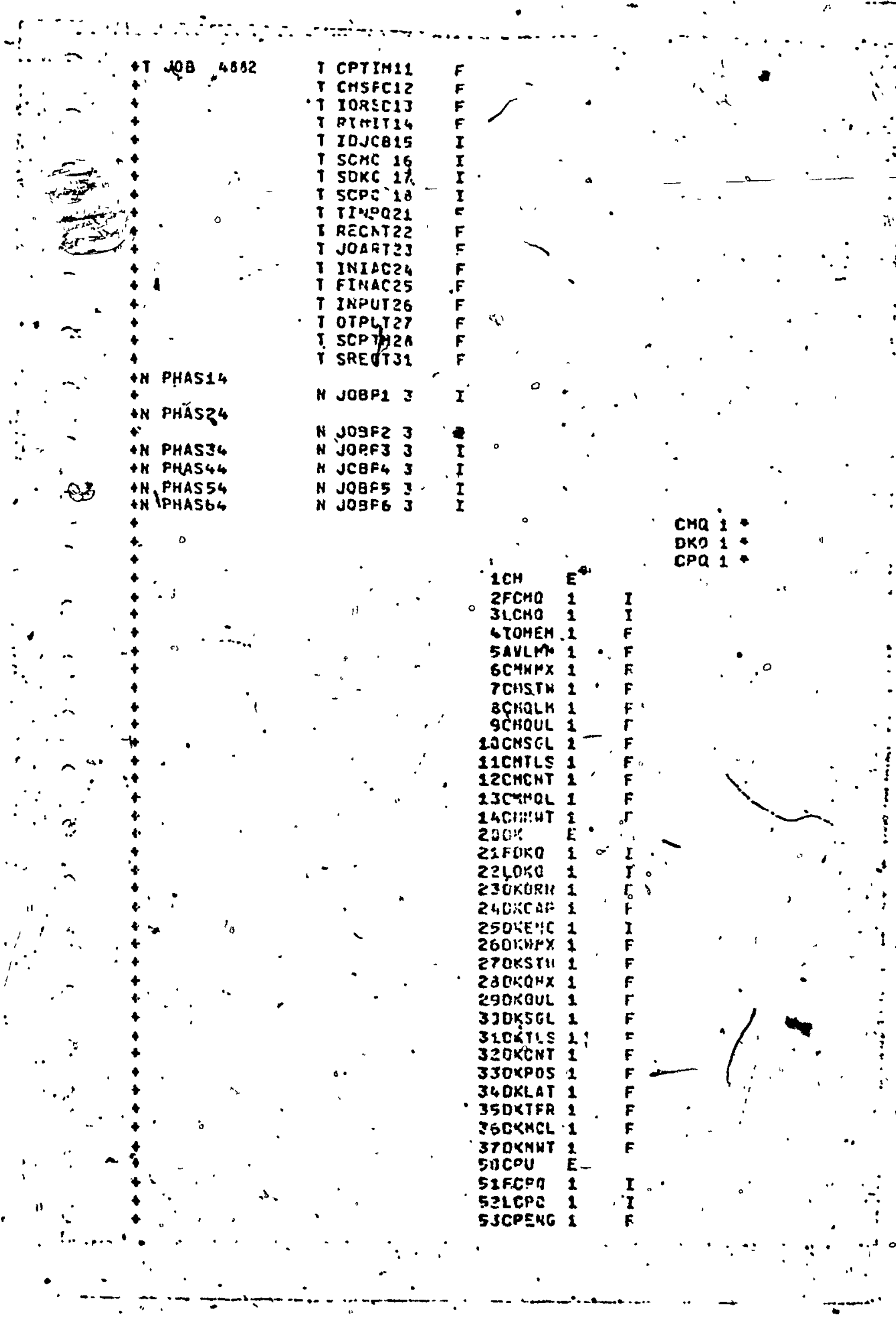


```
SIMULATED SYSTEMS RESULTS.

      SIMULATED TIME=     4.03689
      TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. 1003.

      TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM.  993.
      AVERAGE TRUPUT.  245.98

      C- ANALYSIS

                    CM        DISK      CPU
   C-LENGTH         0.0       0.0       4.0
   MEAN QL          0.043     1.260     1.952
   MAX QL           4.0       18.0      16.0
   MEAN W           0.030     0.002     0.002
   MAX WAIT         0.136     0.027     0.042
   TOTAL JOBS       21.0      7571.0    15214.0
   PERCNT CPU TIME USED                           55.6
   PERCNT I/O ACTIVITY                            36.8
   AVG.  CENTRAL MEM UTY                         146.6
   TOTAL JOBS REJECTD EXCESS MEM DEMAND           0.0
```

λt =14 secs.
CM SIZE 500K

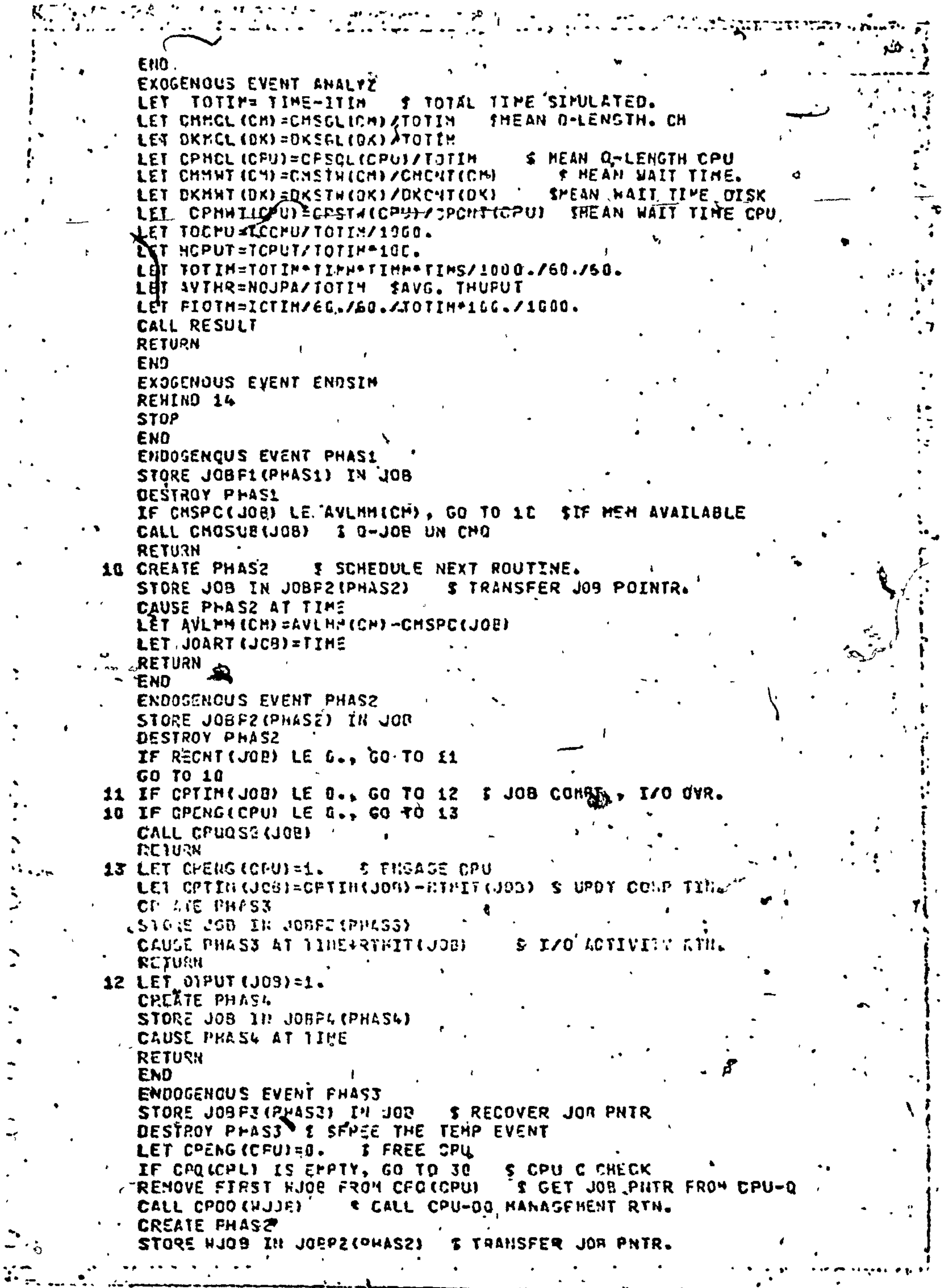
APPENDIX B

APPENDIX B

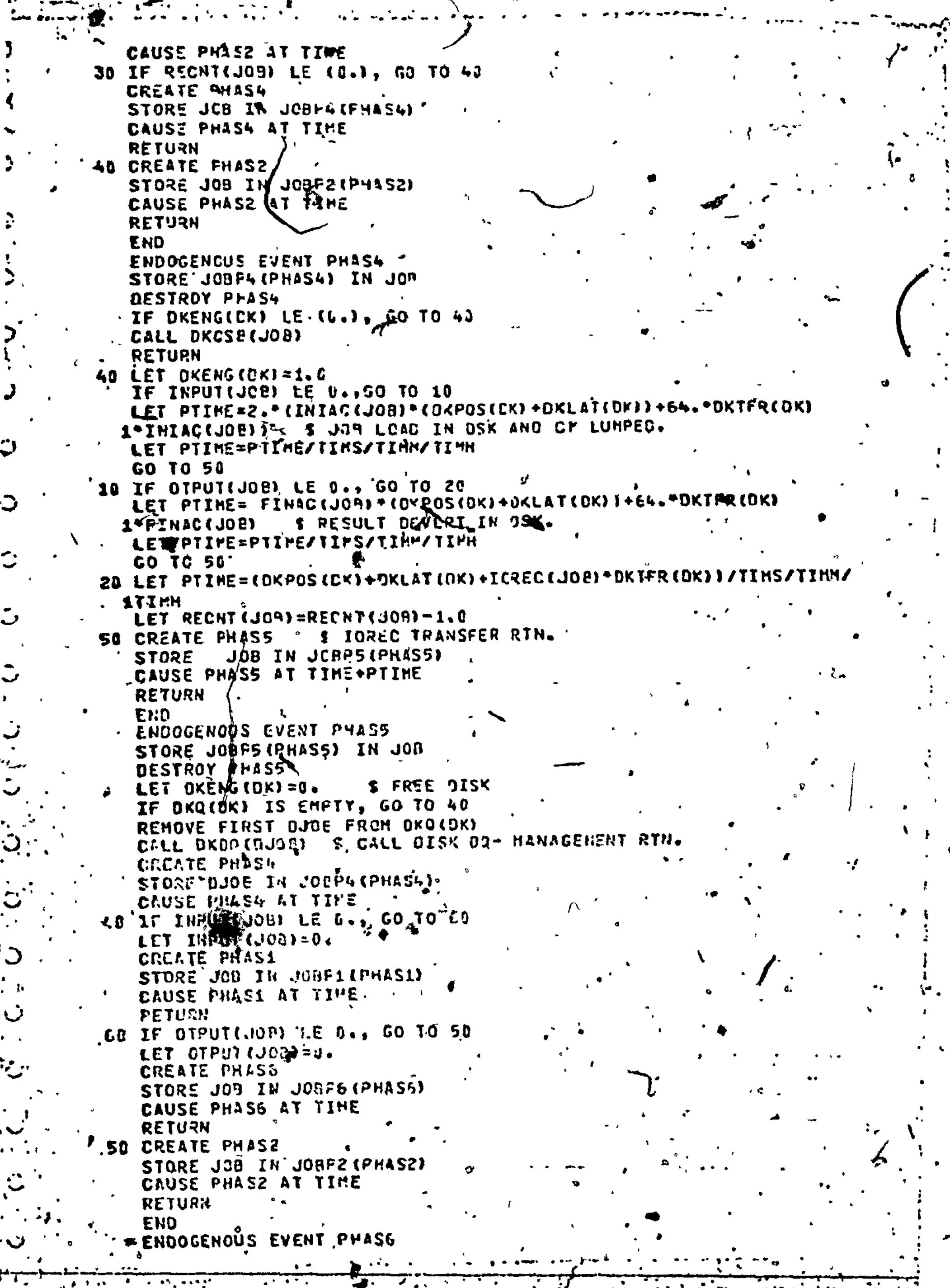


```
+T JOB  4882      T CPTIM11    F
+                 T CMSFC12    F
+                 T IORSC13    F
+                 T RTMIT14    F
+                 T IDJCB15    I
+                 T SCMC  16   I
+                 T SDKC  17   I
+                 T SCPC  18   I
+                 T TINPQ21    F
+                 T RECNT22    F
+                 T JOART23    F
+                 T INIAC24    F
+                 T FINAC25    F
+                 T INPUT26    F
+                 T OTPLT27    F
+                 T SCPTM28    F
+                 T SREQT31    F
+N PHAS14
+                 N JOBP1 3    I
+N PHAS24
+                 N JOBF2 3    I
+N PHAS34         N JOBF3 3    I
+N PHAS44         N JCBF4 3    I
+N PHAS54         N JOBF5 3    I
+N PHAS64         N JOBF6 3    I
+                                                     CMQ 1 +
+                                                     DKQ 1 +
+                                                     CPQ 1 +
+                               1CH     E
+                               2FCMQ   1    I
+                               3LCMQ   1    I
+                               4TOMEM  1    F
+                               5AVLMM  1    F
+                               6CMMPX  1    F
+                               7CMSTM  1    F
+                               8CMQLK  1    F
+                               9CMQUL  1    F
+                              10CMSGL  1    F
+                              11CMTLS  1    F
+                              12CMCNT  1    F
+                              13CMMQL  1    F
+                              14CMNWT  1    F
+                              20DK     E
+                              21FDKQ   1    I
+                              22LDKQ   1    I
+                              23DKQRN  1    F
+                              24DKCAP  1    F
+                              25DKEMC  1    I
+                              26DKMPX  1    F
+                              27DKSTM  1    F
+                              28DKQMX  1    F
+                              29DKQUL  1    F
+                              30DKSGL  1    F
+                              31DKTLS  1    F
+                              32DKCNT  1    F
+                              33DKPOS  1    F
+                              34DKLAT  1    F
+                              35DKTFR  1    F
+                              36DKMCL  1    F
+                              37DKNWT  1    F
+                              50CPU    E
+                              51FCPQ   1    I
+                              52LCPQ   1    I
+                              53CPENG  1    F
```

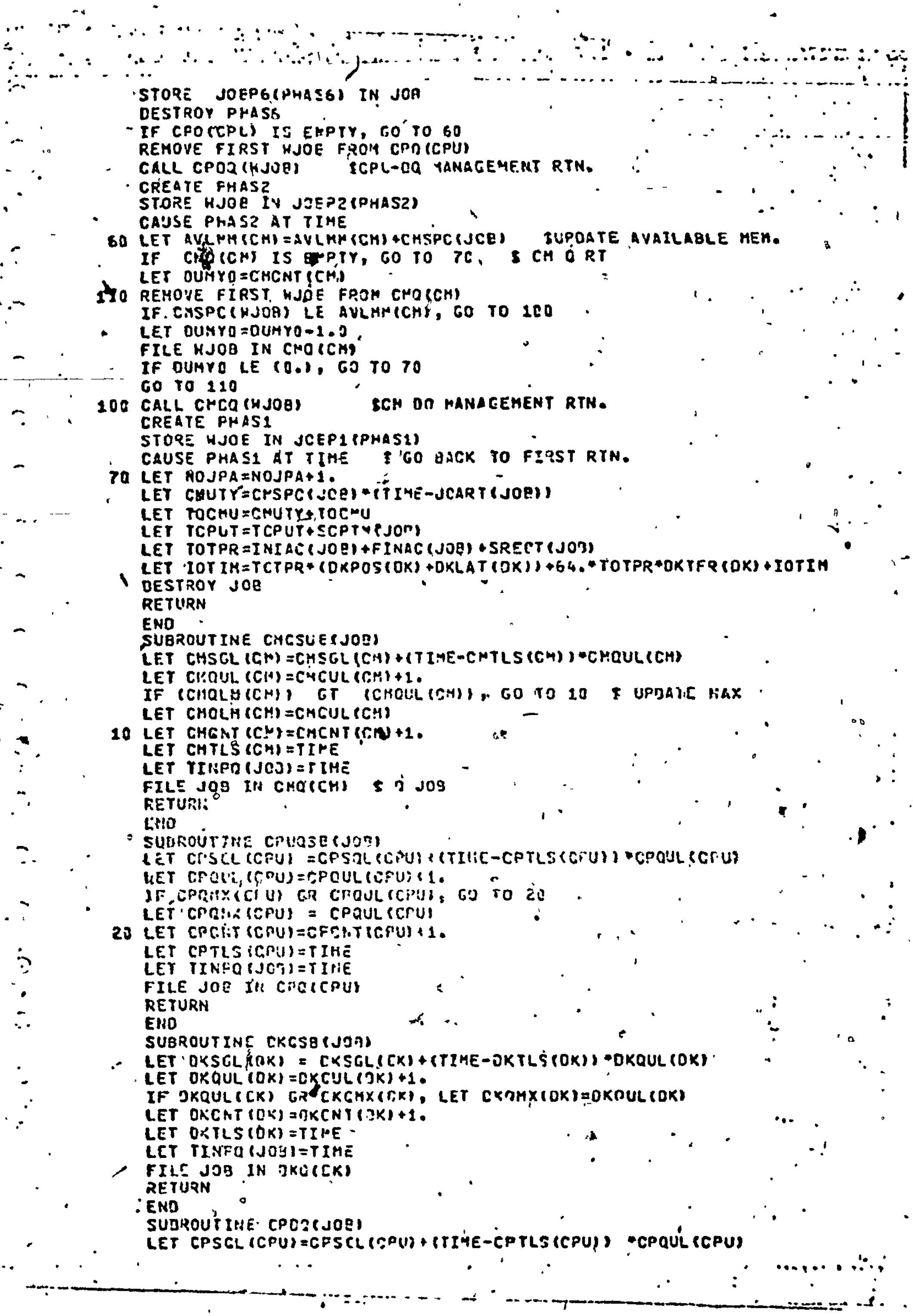
```
    +                                        54CPMMX  1     F
    +                                        55CPSTM  1     F
    +                                        56CPDMX  1     F
    +                                        57CPOUL  1     F
    +                                        58CPSOL  1     F
    +                                        59CPTLS  1     F
    +                                        60CPCNT  1     F
    +                                        61CPMOL  1     F
    +                                        62CPMMT  1     F
    +                                        63NOJOB  0     F
    +                                        64NOJPA  0     F
    +                                        65TOTIM  0     F
    +                                        66AVTMR  0     F
    +                                        67ITIM   0     F
    +                                        68DUMYD  0     F
    +                                        69TIMS   0     FX
    +                                        70TIMM   0     FX
    +                                        71TIMH   0     FX
    +                                        72REJCT  0     F
    +                                        73TCPUT  0     F
    +                                        74TOTPR  0     F
    +                                        75IOTIM  0     F
    +                                        76PIOTM  0     F
    +                                        77TOCMU  0     F
    +                                        78CMUTY  0     F
    +                                        79MCPUT  0     F
             EVENTS
             3 EXOGENOUS
              ARRVL(1)
              ANALYZ(2)
              ENDSIM(3)
             6 ENDOGENOUS
              PHAS1
              PHAS2
              PHAS3
              PHAS4
              PHAS5
              PHAS6
              END
             EXOGENOUS EVENT ARRVL
             CREATE JOB
             LET JOART(JOB)=TIME    $ JOB ARRIVAL TIME
             READ IDJOB(JOB),CPTIM(JOB),CMSPC(JOB),INTAC(JOB),
            1FINAC(JOB),RECNT(JOB)
             FORMAT(I6,D2.4,D3.0,D3.0,D3.0,D3.0)
             LET NOJOB=NOJOB+1.0
             LET CMSPC(JOB)=CMSPC(JOB)*1000.
             IF CMSPC(JOB) GT TCMEM(CM), GO TO 20
             LET CPTIM(JOB)=CPTIM(JOB)*1000./TIMS/TIMM/TIMH   $CONVERT TO DAYS.
             LET SCPTM(JOB)=CPTIM(JOB)
             LET SRECT(JOB)=RECNT(JOB)
             LET IOREC(JOB)=64.
             IF RECNT(JOB) LE 0., GO TO 10
             LET RTMIT(JOB)=CPTIM(JOB)/RECNT(JOB)  $ INTR. REC COMPTE
             GO TO 11
          10 LET RTMIT(JOB)=CPTIM(JOB)
          11 LET INPUT(JOB)=1.
             CREATE PHAS4
             STORE JOB IN JOPF4(PHAS4)
             CAUSE PHAS4 AT TIME
             RETURN
          20 LET REJCT=REJCT+1.
             DESTROY JOB
             RETURN
```



```
    END.
    EXOGENOUS EVENT ANALYZ
    LET  TOTIM= TIME-ITIM    $ TOTAL TIME SIMULATED.
    LET CMMCL(CM)=CMSGL(CM)/TOTIM    $MEAN Q-LENGTH. CH
    LET DKMCL(DK)=DKSGL(DK)/TOTIM
    LET CPMCL(CPU)=CPSQL(CPU)/TOTIM      $ MEAN Q-LENGTH CPU
    LET CMMWT(CM)=CMSTW(CM)/CMCNT(CM)       $ MEAN WAIT TIME.
    LET DKMWT(DK)=DKSTW(DK)/DKCNT(DK)     $MEAN WAIT TIME DISK
    LET  CPMWT(CPU)=CPSTW(CPU)/CPCNT(CPU)   $MEAN WAIT TIME CPU.
    LET TOCPU=TCCPU/TOTIM/1000.
    LET HCPUT=TCPUT/TOTIM*100.
    LET TOTIM=TOTIM*TIMH*TIMM*TIMS/1000./60./60.
    LET AVTHR=NOJPA/TOTIM  $AVG. THUPUT
    LET FIOTM=ICTIM/60./60./TOTIM*100./1000.
    CALL RESULT
    RETURN
    END
    EXOGENOUS EVENT ENDSIM
    REWIND 14
    STOP
    END
    ENDOGENQUS EVENT PHAS1
    STORE JOBF1(PHAS1) IN JOB
    DESTROY PHAS1
    IF CMSPC(JOB) LE AVLMM(CM), GO TO 10  $IF MEM AVAILABLE
    CALL CMQSUB(JOB)  $ Q-JOB ON CMQ
    RETURN
 10 CREATE PHAS2    $ SCHEDULE NEXT ROUTINE.
    STORE JOB IN JOBP2(PHAS2)    $ TRANSFER JOB POINTR.
    CAUSE PHAS2 AT TIME
    LET AVLMM(CM)=AVLMM(CM)-CMSPC(JOB)
    LET JOART(JOB)=TIME
    RETURN
    END
    ENDOGENOUS EVENT PHAS2
    STORE JOBP2(PHAS2) IN JOB
    DESTROY PHAS2
    IF RECNT(JOB) LE 0., GO TO 11
    GO TO 10
 11 IF CPTIM(JOB) LE 0., GO TO 12  $ JOB COMPL, I/O OVR.
 10 IF CPENG(CPU) LE 0., GO TO 13
    CALL CPUQS3(JOB)
    RETURN
 13 LET CPENG(CPU)=1.    $ ENGAGE CPU
    LET CPTIM(JOB)=CPTIM(JOB)-RTWIT(JOB)  $ UPDT COMP TIM.
    CREATE PHAS3
    STORE JOB IN JOBP3(PHAS3)
    CAUSE PHAS3 AT TIME+RTWIT(JOB)     $ I/O ACTIVITY RTN.
    RETURN
 12 LET OTPUT(JOB)=1.
    CREATE PHAS4
    STORE JOB IN JOBP4(PHAS4)
    CAUSE PHAS4 AT TIME
    RETURN
    END
    ENDOGENOUS EVENT PHAS3
    STORE JOBP3(PHAS3) IN JOB    $ RECOVER JOB PNTR
    DESTROY PHAS3  $ SFREE THE TEMP EVENT
    LET CPENG(CPU)=0.    $ FREE CPU
    IF CPQ(CPU) IS EMPTY, GO TO 30    $ CPU Q CHECK
    REMOVE FIRST WJOB FROM CPQ(CPU)    $ GET JOB PNTR FROM CPU-Q
    CALL CPQO(WJOB)     $ CALL CPU-QQ MANAGEMENT RTN.
    CREATE PHAS2
    STORE WJOB IN JOBP2(PHAS2)   $ TRANSFER JOB PNTR.
```

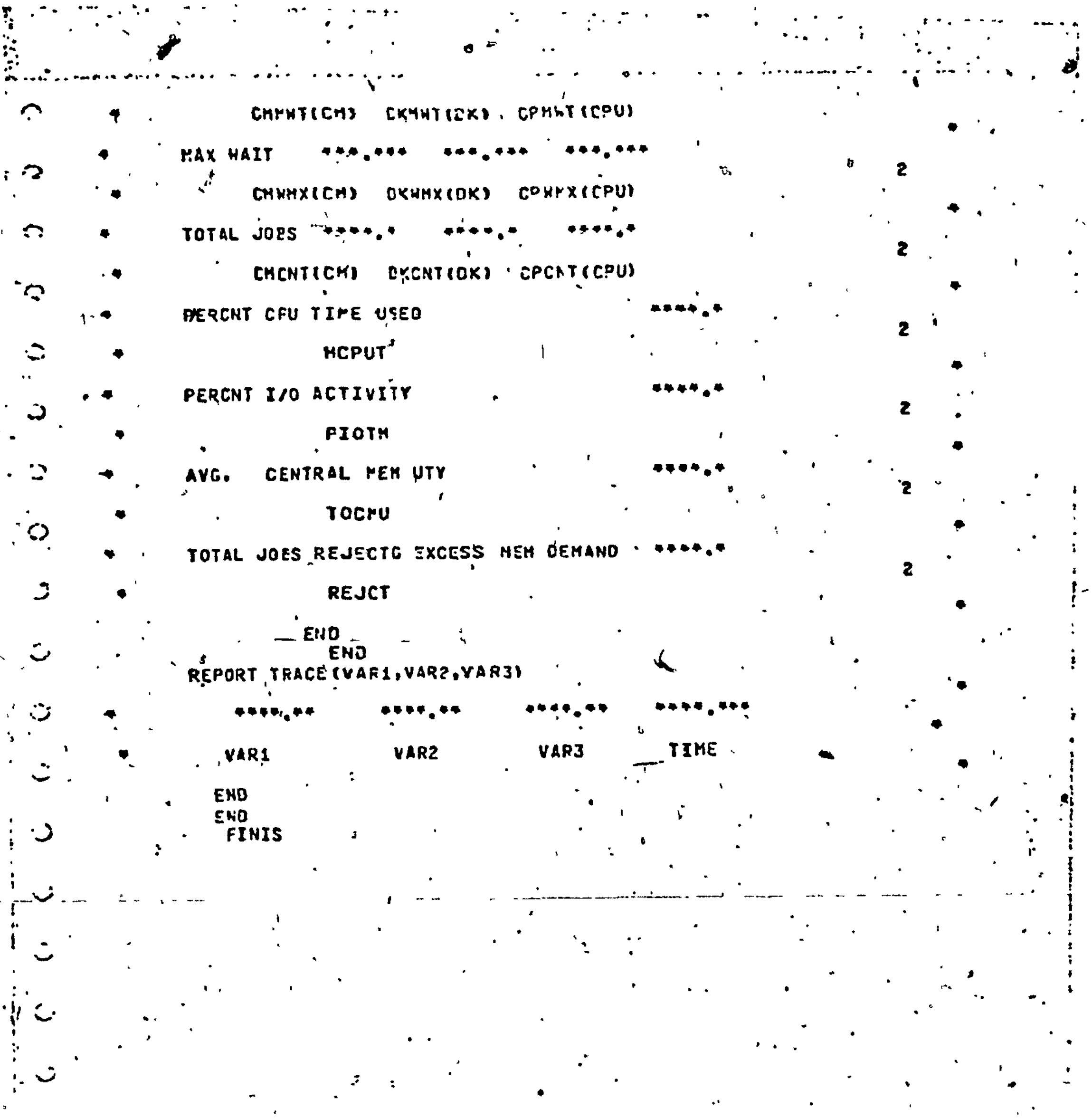


```
       CAUSE PHAS2 AT TIME
    30 IF RECNT(JOB) LE (0.), GO TO 40
       CREATE PHAS4
       STORE JOB IN JOBF4(PHAS4)
       CAUSE PHAS4 AT TIME
       RETURN
    40 CREATE PHAS2
       STORE JOB IN JOBF2(PHAS2)
       CAUSE PHAS2 AT TIME
       RETURN
       END
       ENDOGENOUS EVENT PHAS4
       STORE JOBF4(PHAS4) IN JOB
       DESTROY PHAS4
       IF DKENG(DK) LE (0.), GO TO 40
       CALL DKCSB(JOB)
       RETURN
    40 LET DKENG(DK)=1.0
       IF INPUT(JOB) LE 0.,GO TO 10
       LET PTIME=2.*(INIAC(JOB)*(DKPOS(DK)+DKLAT(DK))+64.*DKTFR(DK)
      1*INIAC(JOB))   $ JOB LOAD IN DSK AND CP LUMPED.
       LET PTIME=PTIME/TIMS/TIMM/TIMH
       GO TO 50
    10 IF OTPUT(JOB) LE 0., GO TO 20
       LET PTIME= FINAC(JOB)*(DKPOS(DK)+DKLAT(DK))+64.*DKTFR(DK)
      1*FINAC(JOB)    $ RESULT DEVERI IN DSK.
       LET PTIME=PTIME/TIMS/TIMM/TIMH
       GO TO 50
    20 LET PTIME=(DKPOS(DK)+DKLAT(DK)+ICREC(JOB)*DKTFR(DK))/TIMS/TIMM/
      1TIMH
       LET RECNT(JOB)=RECNT(JOB)-1.0
    50 CREATE PHAS5     $ IOREC TRANSFER RTN.
       STORE   JOB IN JOBF5(PHAS5)
       CAUSE PHAS5 AT TIME+PTIME
       RETURN
       END
       ENDOGENOUS EVENT PHAS5
       STORE JOBF5(PHAS5) IN JOB
       DESTROY PHAS5
       LET DKENG(DK)=0.      $ FREE DISK
       IF DKQ(DK) IS EMPTY, GO TO 40
       REMOVE FIRST DJOB FROM DKQ(DK)
       CALL DKDQ(DJOB)  $ CALL DISK DQ- MANAGEMENT RTN.
       CREATE PHAS4
       STORE DJOB IN JOBF4(PHAS4)
       CAUSE PHAS4 AT TIME
    40 IF INPUT(JOB) LE 0., GO TO 60
       LET INPUT(JOB)=0.
       CREATE PHAS1
       STORE JOB IN JOBF1(PHAS1)
       CAUSE PHAS1 AT TIME
       RETURN
    60 IF OTPUT(JOB) LE 0., GO TO 50
       LET OTPUT(JOB)=0.
       CREATE PHAS6
       STORE JOB IN JOBF6(PHAS6)
       CAUSE PHAS6 AT TIME
       RETURN
    50 CREATE PHAS2
       STORE JOB IN JOBF2(PHAS2)
       CAUSE PHAS2 AT TIME
       RETURN
       END
      *ENDOGENOUS EVENT PHAS6
```

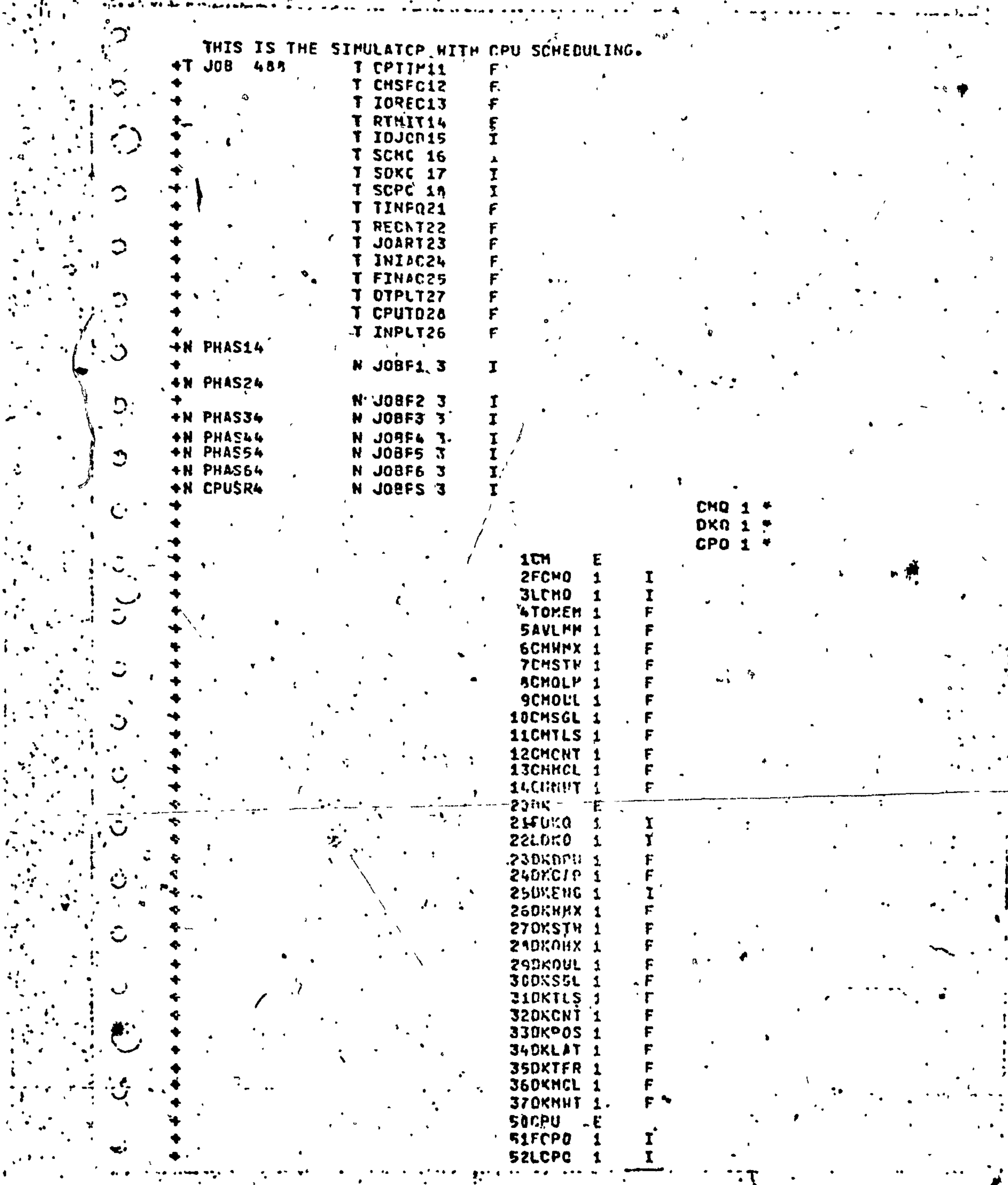


```
    STORE  JOEP6(PHAS6) IN JOB
    DESTROY PHAS6
    IF CPO(CPL) IS EMPTY, GO TO 60
    REMOVE FIRST WJOE FROM CPO(CPU)
    CALL CPOQ(WJOB)     $CPL-OQ MANAGEMENT RTN.
    CREATE PHAS2
    STORE WJOB IN JOEP2(PHAS2)
    CAUSE PHAS2 AT TIME
 60 LET AVLMM(CM)=AVLMM(CM)+CMSPC(JCB)    $UPDATE AVAILABLE MEM.
    IF  CMQ(CM) IS EMPTY, GO TO  70,   $ CM Q RT
    LET DUMYQ=CMCNT(CM)
110 REMOVE FIRST WJOE FROM CMQ(CM)
    IF.CMSPC(WJOB) LE AVLMM(CM), GO TO 100
    LET DUMYQ=DUMYQ-1.0
    FILE WJOB IN CMQ(CM)
    IF DUMYQ LE (0.), GO TO 70
    GO TO 110
100 CALL CMCQ(WJOB)       $CM DQ MANAGEMENT RTN.
    CREATE PHAS1
    STORE WJOE IN JOEP1(PHAS1)
    CAUSE PHAS1 AT TIME   $ GO BACK TO FIRST RTN.
 70 LET NOJPA=NOJPA+1.
    LET CMUTY=CMSPC(JOB)*(TIME-JCART(JOB))
    LET TOCMU=CMUTY+TOCMU
    LET TCPUT=TCPUT+SCPTM(JOB)
    LET TOTPR=INIAC(JOB)+FINAC(JOB)+SRECT(JOB)
    LET IOTIM=TCTPR*(DKPOS(DK)+DKLAT(DK))+64.*TOTPR*DKTFR(DK)+IOTIM
    DESTROY JOB
    RETURN
    END
    SUBROUTINE CMCSUB(JOB)
    LET CMSGL(CM)=CMSGL(CM)+(TIME-CMTLS(CM))*CMQUL(CM)
    LET CMQUL(CM)=CMCUL(CM)+1.
    IF (CMQLM(CM))  GT  (CMQUL(CM)), GO TO 10  $ UPDATE MAX
    LET CMQLM(CM)=CMCUL(CM)
 10 LET CMCNT(CM)=CMCNT(CM)+1.
    LET CMTLS(CM)=TIME
    LET TINPQ(JOB)=TIME
    FILE JOB IN CMQ(CM)  $ Q JOB
    RETURN
    END
    SUBROUTINE CPUQSB(JOB)
    LET CPSCL(CPU) =CPSQL(CPU)+(TIME-CPTLS(CPU))*CPQUL(CPU)
    LET CPQUL(CPU)=CPQUL(CPU)+1.
    IF CPQMX(CPU) GR CPQUL(CPU), GO TO 20
    LET CPQMX(CPU) = CPQUL(CPU)
 20 LET CPCNT(CPU)=CPCNT(CPU)+1.
    LET CPTLS(CPU)=TIME
    LET TINPQ(JOB)=TIME
    FILE JOB IN CPQ(CPU)
    RETURN
    END
    SUBROUTINE DKCSB(JOB)
    LET DKSGL(DK) = DKSGL(DK)+(TIME-DKTLS(DK))*DKQUL(DK)
    LET DKQUL(DK)=DKCUL(DK)+1.
    IF DKQUL(DK) GR DKCMX(DK), LET DKQMX(DK)=DKQUL(DK)
    LET DKCNT(DK)=DKCNT(DK)+1.
    LET DKTLS(DK)=TIME
    LET TINPQ(JOB)=TIME
    FILE JOB IN DKQ(DK)
    RETURN
    END
    SUBROUTINE CPD2(JOB)
    LET CPSCL(CPU)=CPSCL(CPU)+(TIME-CPTLS(CPU)) *CPQUL(CPU)
```

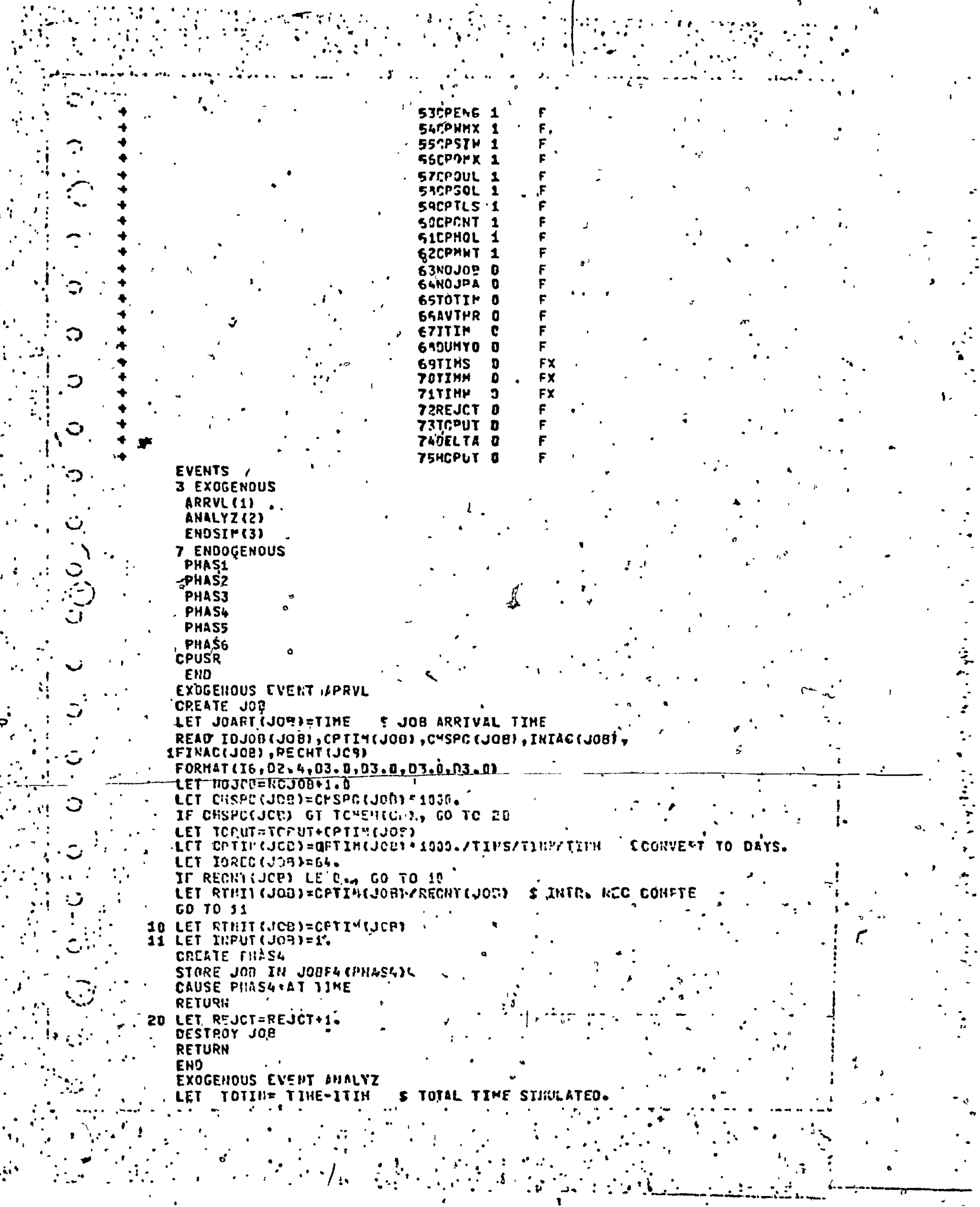
```
         LET CPOUL(CPU)=CPOLL(CPU)-1.
         LET CPSTH(CPU)=CPSTH(CPU)+TIME - TINPO(JOB)
         LET ZAP=TIME-TINPO(JOB)
         IF CPHMX(CPU) GR ZAP , GO TO 30
         LET CPHMX(CPU)= ZAP
      30 LET CPTLS(CPU)=TIME
         RETURN
         END
         SUBROUTINE DKOC(JOB)
         LET DKSGL(DK)= DKSGL(DK)+(TIME-DKTLS(DK))*DKOUL(DK)
         LET DKOUL(DK)=DKOUL(DK)-1.
         LET DKSTH(DK)=DKSTH(DK)+TIME-TINFO(JOB)
         LET TAP=TIME-TINFO(JOB)
         IF TAP GR DKHMX(DK), LET DKHMX(DK)=TAP
         LET DKTLS(DK)=TIME
         RETURN
         END
         SUBROUTINE CMOC(JOB)
         LET CMSGL(CM)=CMSGL(CM)+(TIME-CMTLS(CM))*CMOUL(CM)
         LET CMOUL(CM)=CMOUL(CM)-1.
         LET CMSTH(CM)=CMSTH(CM)+TIME-TINFQ(JOB)
         LET DOP=TIME-TINPC(JOB)
         IF DOP GR CMHMX(CM), LET CMHMX(CM)=DOP
         LET CMTLS(CM)=TIME
         RETURN
         END
                  REPORT RESULT

               SIMULATED SYSTEMS RESULTS                                    3

                  SINLLATED TIME=  ***.*****                                1

                                       TOTIM

                  TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. ****.            2
                                                               NOJOB

                  TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM: ****.      1
                                                                     NOJPA

                  AVERAGE TRUPUT. ****.**                                   3

                                      AVTHR

                  Q--ANALYSIS                                               3

                      CM          DISK        CPU                           1

       Q-LENGTH     ****.*      ****.*      ****.*                          2

               CMOUL(CM)   DKOUL(DK)   CPOLL(CPU)

       MEAN QL      ***.***     ***.***     ***.***                         2

               CMMQL(CM)   DKMQL(DK)   CPMQL(CPU)

       MAX QL       ****.*      ****.*      ****.*                          2

              CMQLM(CM)  DKQMX(DK)    CPQMX(CPU)

       MEAN W       ***.***     ***.***     ***.***                         2
```



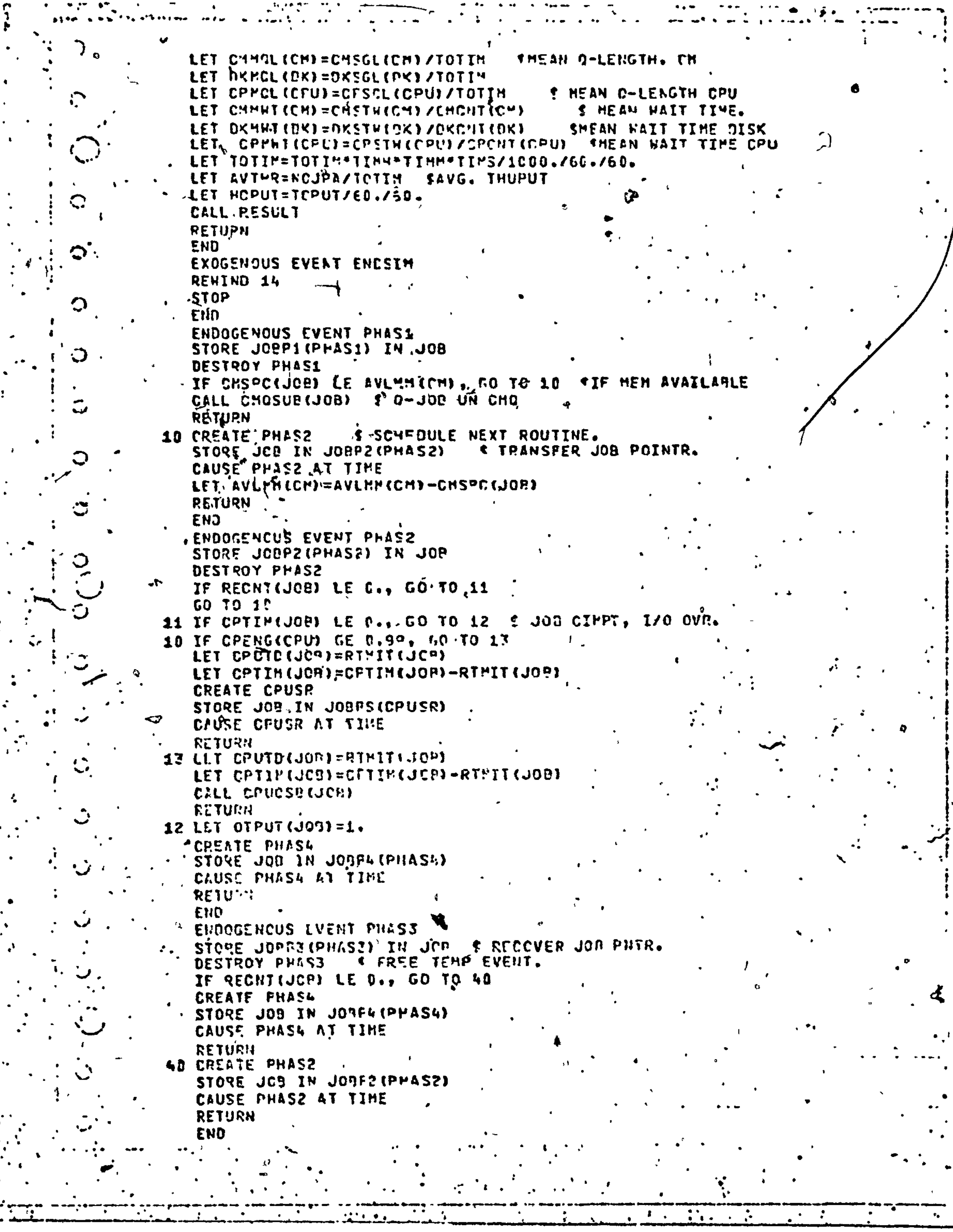
```
         *          CMMWT(CM)   DKMWT(DK)   CPMWT(CPU)
         *      MAX WAIT    ***.***    ***.***    ***.***
         *          CMWMX(CM)   DKWMX(DK)   CPWMX(CPU)              2
         *      TOTAL JOBS  ****.*     ****.*     ****.*
         *          CMCNT(CM)   DKCNT(DK)   CPCNT(CPU)              2
         *      PERCNT CPU TIME USED                      ****.*
         *                  MCPUT                                   2
         *      PERCNT I/O ACTIVITY                       ****.*
         *                  PIOTM                                   2
         *      AVG.   CENTRAL MEM UTY                    ****.*
         *                  TOCMU                                   2
         *      TOTAL JOBS REJECTD EXCESS MEM DEMAND      ****.*
         *                  REJCT                                   2
                      END
                        END
                REPORT TRACE(VAR1,VAR2,VAR3)
         *         ****.**     ****.**     ****.**     ****.***
         *          VAR1        VAR2        VAR3        TIME
                  END
                  END
                   FINIS
```



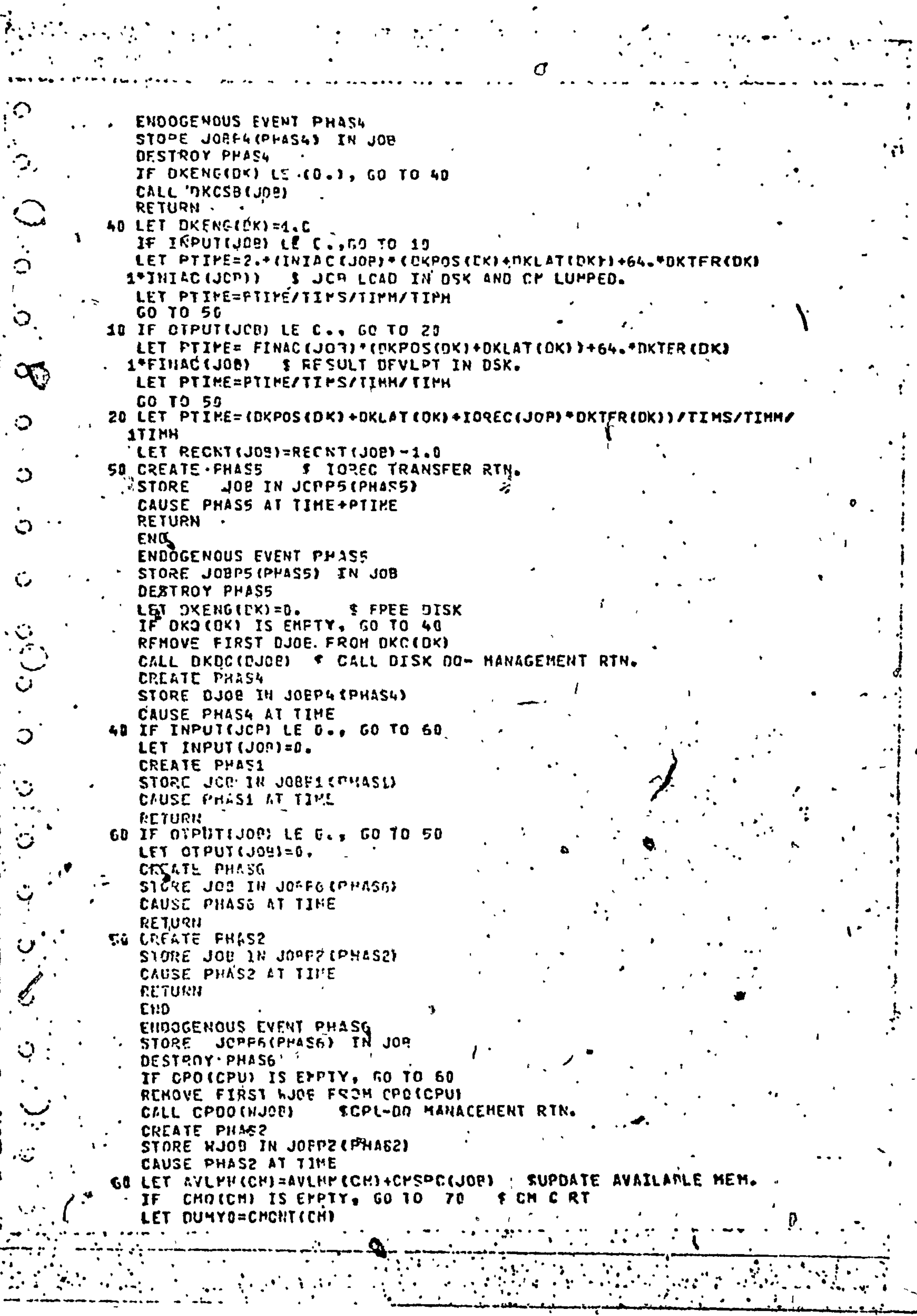
```
    THIS IS THE SIMULATCP WITH CPU SCHEDULING.
+T JOB  485           T CPTIM11     F
+                     T CHSFC12     F
+                     T IOREC13     F
+                     T RTMIT14     F
+                     T IDJCD15     I
+                     T SCMC 16     I
+                     T SDKC 17     I
+                     T SCPC 18     I
+                     T TINFQ21     F
+                     T RECNT22     F
+                     T JOART23     F
+                     T INIAC24     F
+                     T FINAC25     F
+                     T DTPLT27     F
+                     T CPUTD28     F
+                     T INPLT26     F
+N PHAS14
+                     N JOBF1 3     I
+N PHAS24
+                     N JOBF2 3     I
+N PHAS34             N JOBF3 3     I
+N PHAS44             N JOBF4 3     I
+N PHAS54             N JOBF5 3     I
+N PHAS64             N JOBF6 3     I
+N CPUSR4             N JOBFS 3     I
+                                                            CMQ 1 *
+                                                            DKQ 1 *
+                                                            CPO 1 *
+                                         1CM    E
+                                         2FCMO  1     I
+                                         3LCMO  1     I
+                                         4TOMEM 1     F
+                                         5AVLMM 1     F
+                                         6CMHMX 1     F
+                                         7CMSTH 1     F
+                                         8CMOLP 1     F
+                                         9CMOUL 1     F
+                                        10CMSGL 1     F
+                                        11CMTLS 1     F
+                                        12CMCNT 1     F
+                                        13CMMCL 1     F
+                                        14CMMUT 1     F
+                                        20DK    E
+                                        21FDKQ  1     I
+                                        22LDKQ  1     I
+                                        23DKDPU 1     F
+                                        24DKC/P 1     F
+                                        25DKENG 1     I
+                                        26DKHMX 1     F
+                                        27DKSTH 1     F
+                                        28DKOHX 1     F
+                                        29DKOUL 1     F
+                                        30DKSGL 1     F
+                                        31DKTLS 1     F
+                                        32DKCNT 1     F
+                                        33DKPOS 1     F
+                                        34DKLAT 1     F
+                                        35DKTFR 1     F
+                                        36DKMCL 1     F
+                                        37DKMHT 1     F
+                                        50CPU   E
+                                        51FCPQ  1     I
+                                        52LCPQ  1     I
```



```
 +                                53CPENG 1    F
 +                                54CPHMX 1    F,
 +                                55CPSTM 1    F
 +                                56CPOMX 1    F
 +                                57CPOUL 1    F
 +                                58CPSOL 1    F
 +                                59CPTLS 1    F
 +                                60CPCNT 1    F
 +                                61CPHOL 1    F
 +                                62CPMNT 1    F
 +                                63NOJOB 0    F
 +                                64NOJPA 0    F
 +                                65TOTIM 0    F
 +                                66AVTPR 0    F
 +                                67ITIM  C    F
 +                                68DUMYO 0    F
 +                                69TIMS  D    FX
 +                                70TIMM  D    FX
 +                                71TIMP  D    FX
 +                                72REJCT 0    F
 +                                73TCPUT 0    F
 +                                74DELTA 0    F
 +                                75HCPUT 0    F
       EVENTS /
       3 EXOGENOUS
        ARRVL(1)
        ANALYZ(2)
        ENDSIM(3)
       7 ENDOGENOUS
        PHAS1
        PHAS2
        PHAS3
        PHAS4
        PHAS5
        PHAS6
       CPUSR
        END
       EXOGENOUS EVENT ARRVL
       CREATE JOB
       LET JOART(JOB)=TIME    $ JOB ARRIVAL TIME
       READ IDJOB(JOB),CPTIM(JOB),CMSPC(JOB),INIAC(JOB),
      1FINAC(JOB),RECNT(JOB)
       FORMAT(I6,D2.4,D3.0,D3.0,D3.0,D3.0)
       LET NOJOB=NOJOB+1.0
       LET CMSPC(JOB)=CMSPC(JOB)*1000.
       IF CMSPC(JOB) GT TCMEM(C..), GO TO 20
       LET TCPUT=TCPUT+CPTIM(JOB)
       LET CPTIM(JOB)=CPTIM(JOB)*1000./TIMS/TIMM/TIMH   $CONVERT TO DAYS.
       LET IOREC(JOB)=64.
       IF RECNT(JOB) LE 0., GO TO 10
       LET RTMIT(JOB)=CPTIM(JOB)/RECNT(JOB)  $ INTR. REC COMPTE
       GO TO 11
    10 LET RTMIT(JOB)=CPTIM(JOB)
    11 LET INPUT(JOB)=1.
       CREATE PHAS4
       STORE JOB IN JOBF4(PHAS4)
       CAUSE PHAS4 AT TIME
       RETURN
    20 LET REJCT=REJCT+1.
       DESTROY JOB
       RETURN
       END
       EXOGENOUS EVENT ANALYZ
       LET  TOTIM= TIME-ITIM    $ TOTAL TIME SIMULATED.
```

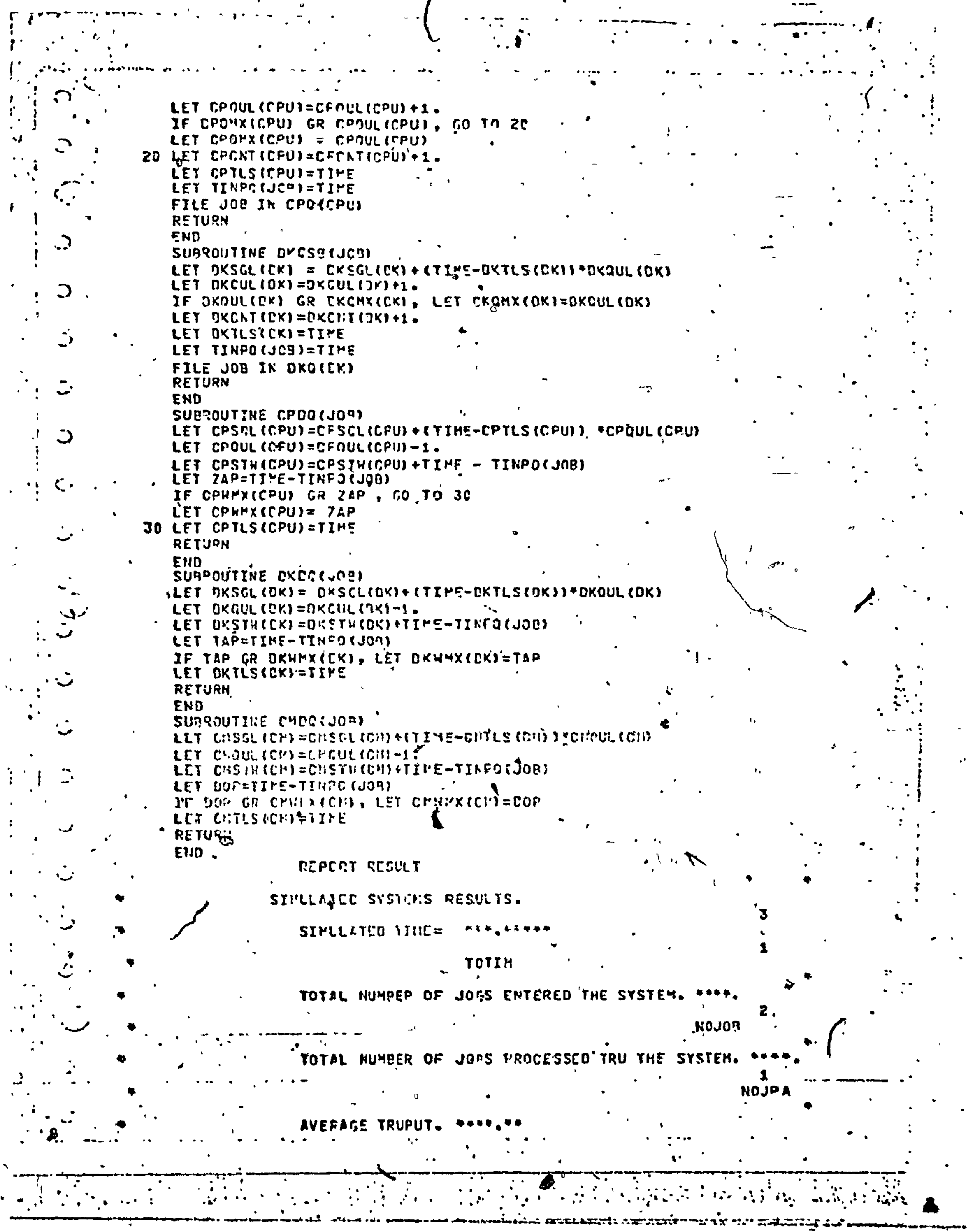


```
    LET CMMQL(CM)=CMSQL(CM)/TOTIM    $MEAN Q-LENGTH, CM
    LET DKMQL(DK)=DKSQL(DK)/TOTIM
    LET CPMQL(CPU)=CPSQL(CPU)/TOTIM     $ MEAN Q-LENGTH CPU
    LET CMMWT(CM)=CMSTW(CM)/CMCNT(CM)      $ MEAN WAIT TIME.
    LET DKMWT(DK)=DKSTW(DK)/DKCNT(DK)     $MEAN WAIT TIME DISK
    LET  CPMWT(CPU)=CPSTW(CPU)/CPCNT(CPU)   $MEAN WAIT TIME CPU
    LET TOTIM=TOTIM*TIMH*TIMM*TIMS/1000./60./60.
    LET AVTHR=NOJPA/TOTIM  $AVG. THUPUT
    LET HCPUT=TCPUT/60./60.
    CALL RESULT
    RETURN
    END
    EXOGENOUS EVENT ENDSIM
    REWIND 14
    STOP
    END
    ENDOGENOUS EVENT PHAS1
    STORE JOBP1(PHAS1) IN JOB
    DESTROY PHAS1
    IF CMSPC(JOB) LE AVLMM(CM), GO TO 10  $IF MEM AVAILABLE
    CALL CMQSUB(JOB)  $ Q-JOB ON CMQ
    RETURN
 10 CREATE PHAS2     $ SCHEDULE NEXT ROUTINE.
    STORE JOB IN JOBP2(PHAS2)    $ TRANSFER JOB POINTR.
    CAUSE PHAS2 AT TIME
    LET AVLMM(CM)=AVLMM(CM)-CMSPC(JOB)
    RETURN
    END
    ENDOGENOUS EVENT PHAS2
    STORE JOBP2(PHAS2) IN JOB
    DESTROY PHAS2
    IF RECNT(JOB) LE 0., GO TO 11
    GO TO 10
 11 IF CPTIM(JOB) LE 0., GO TO 12  $ JOB CMPT, I/O OVR.
 10 IF CPENG(CPU) GE 0.99, GO TO 13
    LET CPUTD(JOB)=RTMIT(JOB)
    LET CPTIM(JOB)=CPTIM(JOB)-RTMIT(JOB)
    CREATE CPUSR
    STORE JOB IN JOBPS(CPUSR)
    CAUSE CPUSR AT TIME
    RETURN
 13 LET CPUTD(JOB)=RTMIT(JOB)
    LET CPTIM(JOB)=CPTIM(JOB)-RTMIT(JOB)
    CALL CPUQSB(JOB)
    RETURN
 12 LET OTPUT(JOB)=1.
    CREATE PHAS4
    STORE JOB IN JOBP4(PHAS4)
    CAUSE PHAS4 AT TIME
    RETURN
    END
    ENDOGENOUS EVENT PHAS3
    STORE JOBP3(PHAS3) IN JOB   $ RECOVER JOB PNTR.
    DESTROY PHAS3    $ FREE TEMP EVENT.
    IF RECNT(JOB) LE 0., GO TO 40
    CREATE PHAS4
    STORE JOB IN JOBP4(PHAS4)
    CAUSE PHAS4 AT TIME
    RETURN
 40 CREATE PHAS2
    STORE JOB IN JOBP2(PHAS2)
    CAUSE PHAS2 AT TIME
    RETURN
    END
```

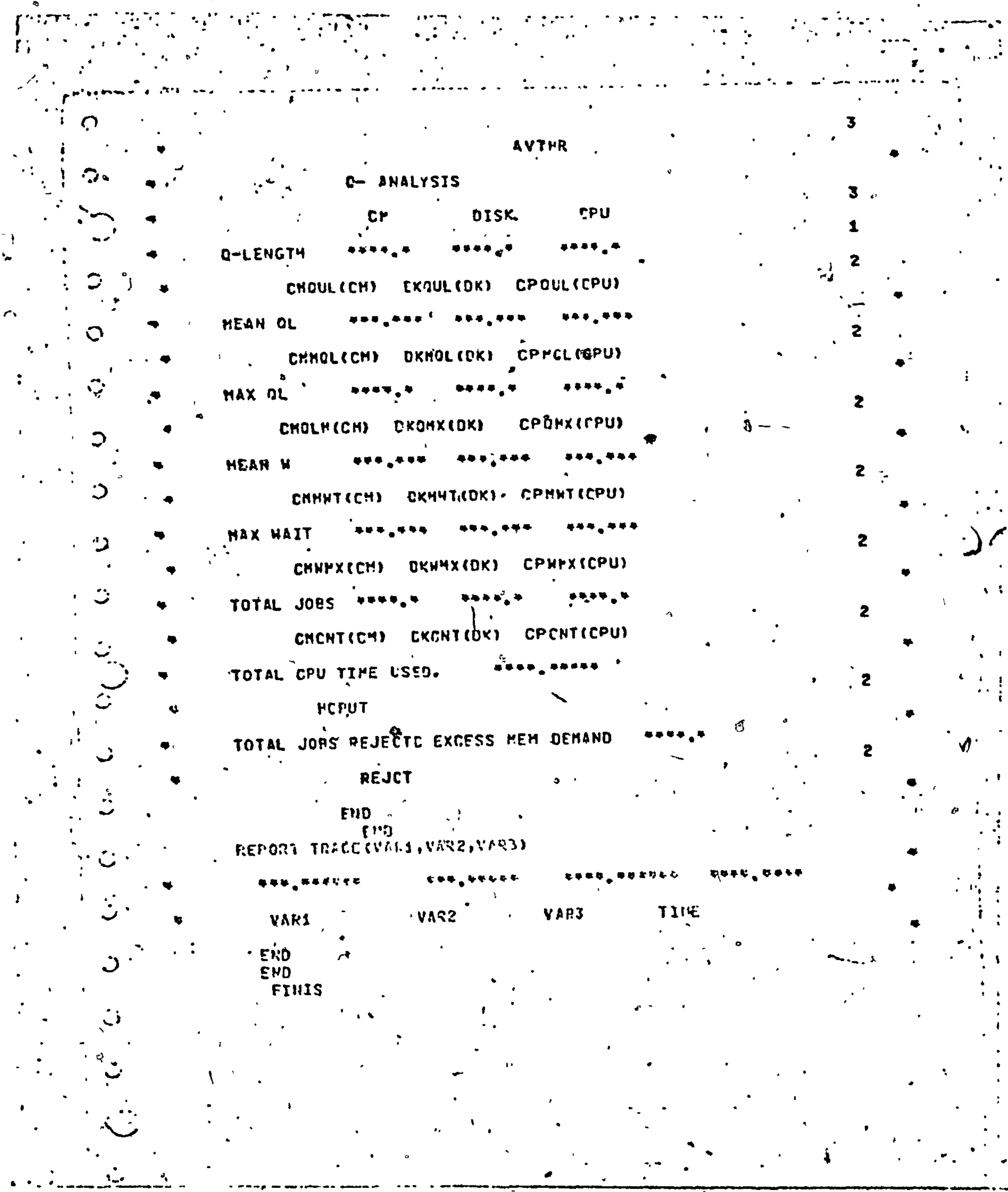


```
       ENDOGENOUS EVENT PHAS4
       STOPE JOBF4(PHAS4) IN JOB
       DESTROY PHAS4
       IF DKENG(DK) LE (0.), GO TO 40
       CALL DKCSB(JOB)
       RETURN
    40 LET DKENG(DK)=1.C
       IF INPUT(JOB) LE C.,GO TO 10
       LET PTIPE=2.*(INIAC(JOP)*(DKPOS(DK)+DKLAT(DK))+64.*DKTFR(DK)
      1*INIAC(JCP))    $ JCB LCAD IN DSK AND CP LUMPED.
       LET PTIPE=PTIPE/TIPS/TIPM/TIPH
       GO TO 50
    10 IF OTPUT(JCB) LE C., GO TO 20
       LET PTIPE= FINAC(JOB)*(DKPOS(DK)+DKLAT(DK))+64.*DKTER(DK)
      1*FINAC(JOB)    $ RESULT DFVLPT IN DSK.
       LET PTIME=PTIPE/TIPS/TIMH/TIPH
       GO TO 50
    20 LET PTIME=(DKPOS(DK)+DKLAT(DK)+IOREC(JOP)*DKTFR(DK))/TIMS/TIMM/
      1TIMH
       LET RECNT(JOB)=RECNT(JOB)-1.0
    50 CREATE PHAS5    $ IOREC TRANSFER RTN.
       STORE   JOB IN JCPP5(PHAS5)
       CAUSE PHAS5 AT TIME+PTIME
       RETURN
       END
       ENDOGENOUS EVENT PHAS5
       STORE JOBP5(PHAS5) IN JOB
       DESTROY PHAS5
       LET DKENG(DK)=0.     $ FPEE DISK
       IF DKQ(DK) IS EMPTY, GO TO 40
       REMOVE FIRST DJOB FROM DKQ(DK)
       CALL DKDQ(DJOB)  $ CALL DISK DQ- MANAGEMENT RTN.
       CREATE PHAS4
       STORE DJOB IN JOBP4(PHAS4)
       CAUSE PHAS4 AT TIME
    40 IF INPUT(JCP) LE 0., GO TO 60
       LET INPUT(JOP)=0.
       CREATE PHAS1
       STORE JOB IN JOBP1(PHAS1)
       CAUSE PHAS1 AT TIME
       RETURN
    60 IF OTPUT(JOP) LE 0., GO TO 50
       LET OTPUT(JOB)=0.
       CREATE PHAS6
       STORE JOB IN JOPF6(PHAS6)
       CAUSE PHAS6 AT TIME
       RETURN
    50 CREATE PHAS2
       STORE JOB IN JOPP2(PHAS2)
       CAUSE PHAS2 AT TIME
       RETURN
       END
       ENDOGENOUS EVENT PHAS6
       STORE  JOPP6(PHAS6) IN JOB
       DESTROY PHAS6
       IF CPQ(CPU) IS EMPTY, GO TO 60
       REMOVE FIRST WJOB FROM CPQ(CPU)
       CALL CPDQ(WJOB)     $CPU-DQ MANAGEMENT RTN.
       CREATE PHAS2
       STORE WJOB IN JOBP2(PHAS2)
       CAUSE PHAS2 AT TIME
    60 LET AVLMM(CM)=AVLMM(CM)+CMSPC(JOB)  $UPDATE AVAILABLE MEM.
       IF  CMQ(CM) IS EMPTY, GO TO  70   $ CM Q RT
       LET DUMYO=CMCNT(CM)
```

```
 110 REMOVE FIRST WJOB FROM CMQ(CM)
     IF CMSPC(WJOB) LE AVLMM(CM), GO TO 100
     LET DUMYO=DUMYO-1.
     FILE WJOB IN CMQ(CM)
     IF DUMYO LE (0.), GO TO 70
     GO TO 110
 100 CALL CMDQ(WJOB)          $CM DQ MANAGEMENT RTN.
     CREATE PHAS1
     STORE WJOB IN JOBP1(PHAS1)
     CAUSE PHAS1 AT TIME    $ GO BACK TO FIRST RTN.
  70 LET NOJPA=NOJPA+1.
     DESTROY JOB
     RETURN
     END
     ENDOGENOUS EVENT CPUSR
     STORE JOBFS(CPUSR) IN JOB
     DESTROY CPUSR
     LET CPENG(CPU)=0.   $ FREE CPU.
     IF CPUTD(JOB) LE 0., GO TO 10
     IF CPQ(CPU) IS EMPTY, GO TO 20
     CALL CPUQSE(JOB)
     REMOVE FIRST WJOB FROM CPQ(CPU)   $ GET JOB PNTR.
     CALL CPDQ(WJOB)
  60 STORE WJOB IN JOB
  20 IF CPUTD(JOB) LE DELTA, GO TO  30
     LET CPUTD(JOB)=CPUTD(JOB)-DELTA
     LET CPENG(CPU)=1.
     CREATE CPUSR
     STORE JOB IN JOBFS(CPUSR)
     CAUSE CPUSR AT TIME+DELTA
     RETURN
  10 IF RECNT(JOB) LE 0.,GO TO 40
     CREATE PHAS3
     STORE JOB IN JOBP3(PHAS3)
     CAUSE PHAS3 AT TIME
  70 IF CPQ(CPU) IS EMPTY, GO TO 50
     REMOVE FIRST WJOB FROM CPQ(CPU)
     CALL CPDQ(WJOB)
     GO TO 60
  30 LET CPENG(CPU)=1.
     CREATE CPUSR
     STORE JOB IN JOBFS(CPUSR)
     CAUSE CPUSR AT TIME+CPUTD(JOB)
     LET CPUTD(JOB)=0.
     RETURN
  40 CREATE PHAS2
     STORE JOB IN JOBP2(PHAS2)
     CAUSE PHAS2 AT TIME
     GO TO 70
  50 RETURN
     END
     SUBROUTINE CMQSUE(JOB)
     LET CMSQL(CM)=CMSQL(CM)+(TIME-CMTLS(CM))*CMQUL(CM)
     LET CMQUL(CM)=CMQUL(CM)+1.
     IF (CMQLM(CM))  GT  (CMQUL(CM)), GO TO 10   $ UPDATE MAX
     LET CMQLM(CM)=CMQUL(CM)
  10 LET CMCNT(CM)=CMCNT(CM)+1.
     LET CMTLS(CM)=TIME
     LET TINFC(JOB)=TIME
     FILE JOB IN CMQ(CM)    $ Q JOB
     RETURN
     END
     SUBROUTINE CPUQSE(JOB)
     LET CPSQL(CPU) =CPSQL(CPU)+(TIME-CPTLS(CPU))*CPQUL(CPU)
```



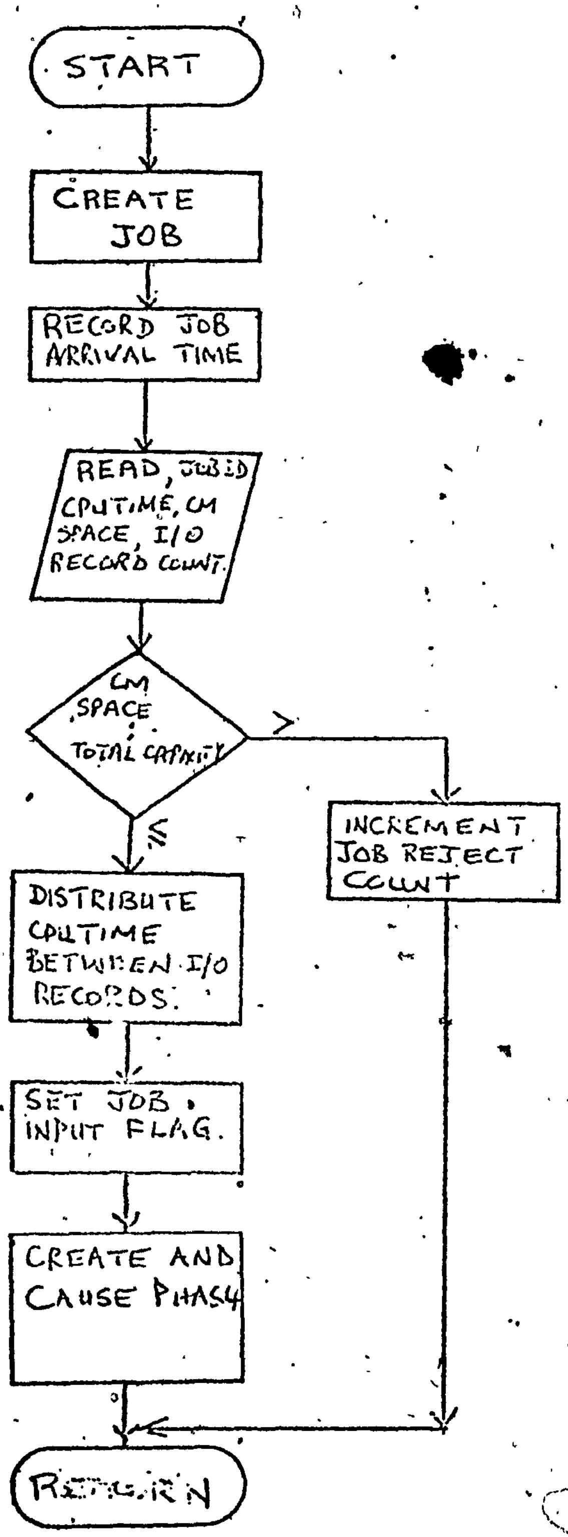
```
       LET CPQUL(CPU)=CPQUL(CPU)+1.
       IF CPQMX(CPU) GR CPQUL(CPU), GO TO 20
       LET CPQMX(CPU) = CPQUL(CPU)
    20 LET CPCNT(CPU)=CPCNT(CPU)+1.
       LET CPTLS(CPU)=TIME
       LET TINPQ(JOB)=TIME
       FILE JOB IN CPQ(CPU)
       RETURN
       END
       SUBROUTINE DKCSB(JOB)
       LET DKSGL(DK) = DKSGL(DK)+(TIME-DKTLS(DK))*DKQUL(DK)
       LET DKQUL(DK)=DKQUL(DK)+1.
       IF DKQUL(DK) GR DKQMX(DK), LET DKQMX(DK)=DKQUL(DK)
       LET DKCNT(DK)=DKCNT(DK)+1.
       LET DKTLS(DK)=TIME
       LET TINPQ(JOB)=TIME
       FILE JOB IN DKQ(DK)
       RETURN
       END
       SUBROUTINE CPDQ(JOB)
       LET CPSGL(CPU)=CPSGL(CPU)+(TIME-CPTLS(CPU)) *CPQUL(CPU)
       LET CPQUL(CPU)=CPQUL(CPU)-1.
       LET CPSTW(CPU)=CPSTW(CPU)+TIME - TINPQ(JOB)
       LET ZAP=TIME-TINPQ(JOB)
       IF CPWMX(CPU) GR ZAP , GO TO 30
       LET CPWMX(CPU)= ZAP
    30 LET CPTLS(CPU)=TIME
       RETURN
       END
       SUBROUTINE DKDQ(JOB)
       LET DKSGL(DK)= DKSGL(DK)+(TIME-DKTLS(DK))*DKQUL(DK)
       LET DKQUL(DK)=DKQUL(DK)-1.
       LET DKSTW(DK)=DKSTW(DK)+TIME-TINPQ(JOB)
       LET TAP=TIME-TINPQ(JOB)
       IF TAP GR DKWMX(DK), LET DKWMX(DK)=TAP
       LET DKTLS(DK)=TIME
       RETURN
       END
       SUBROUTINE CMDQ(JOB)
       LET CMSGL(CM)=CMSGL(CM)+(TIME-CMTLS(CM))*CMQUL(CM)
       LET CMQUL(CM)=CMQUL(CM)-1.
       LET CMSTW(CM)=CMSTW(CM)+TIME-TINPQ(JOB)
       LET DOP=TIME-TINPQ(JOB)
       IF DOP GR CMWMX(CM), LET CMWMX(CM)=DOP
       LET CMTLS(CM)=TIME
       RETURN
       END .
                    REPORT RESULT

                 SIMULATED SYSTEMS RESULTS.
                                                                3
                    SIMULATED TIME=   ***.*****                 .
                                                                1
                                         TOTIM

                    TOTAL NUMBER OF JOBS ENTERED THE SYSTEM. ****.
                                                                2.
                                                            NOJOB

                    TOTAL NUMBER OF JOBS PROCESSED TRU THE SYSTEM. ****.
                                                                1
                                                               NOJPA

                    AVERAGE TRUPUT. ****.**
```



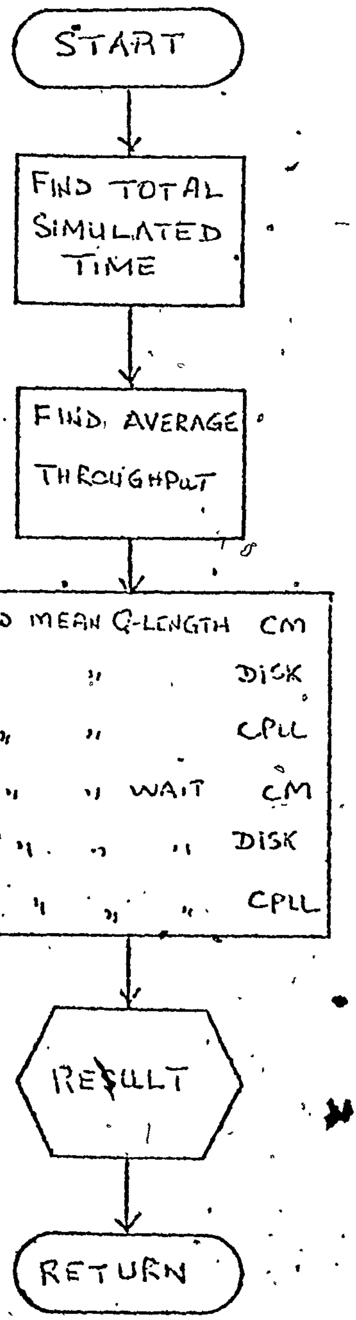
```
                                   AVTHR                                 3

               Q- ANALYSIS                                               3
                       CM          DISK        CPU                       1
  Q-LENGTH     ****.*      ****.*      ****.*                            2
           CMQUL(CM)   DKQUL(DK)   CPQUL(CPU)
  MEAN QL      ***.***     ***.***     ***.***                           2
           CMMQL(CM)   DKMQL(DK)   CPMQL(CPU)
  MAX QL       ****.*      ****.*      ****.*                            2
           CMQLM(CM)   DKQMX(DK)   CPQMX(CPU)
  MEAN W       ***.***     ***.***     ***.***                           2
           CMMWT(CM)   DKMWT(DK)   CPMWT(CPU)
  MAX WAIT     ***.***     ***.***     ***.***                           2
           CMWMX(CM)   DKWMX(DK)   CPWMX(CPU)
  TOTAL JOBS   ****.*      ****.*      ****.*                            2
           CMCNT(CM)   DKCNT(DK)   CPCNT(CPU)
  TOTAL CPU TIME USED.          ****.*****                               2
             MCPUT
  TOTAL JOBS REJECTC EXCESS MEM DEMAND     ****.*                        2
                REJCT
              END
                END
  REPORT TRACE(VAR1,VAR2,VAR3)

     ***.******      ***.*****      ****.******    ****.****

     VAR1            VAR2           VAR3           TIME

   END
   END
    FINIS
```

APPENDIX C

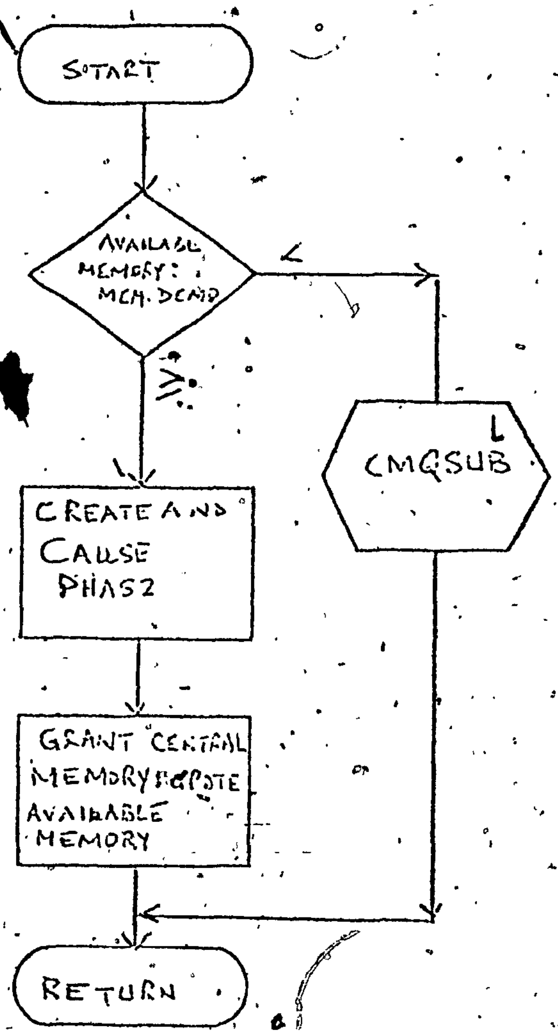
EXOGENOUS EVENT ARRVL


START
CREATE JOB
RECORD JOB ARRIVAL TIME
READ, JOBID CPUTIME, CM SPACE, I/O RECORD COUNT.
CM SPACE : TOTAL CAPACITY
>
≤
INCREMENT JOB REJECT COUNT
DISTRIBUTE CPUTIME BETWEEN I/O RECORDS.
SET JOB INPUT FLAG.
CREATE AND CAUSE PHAS4
RETURN

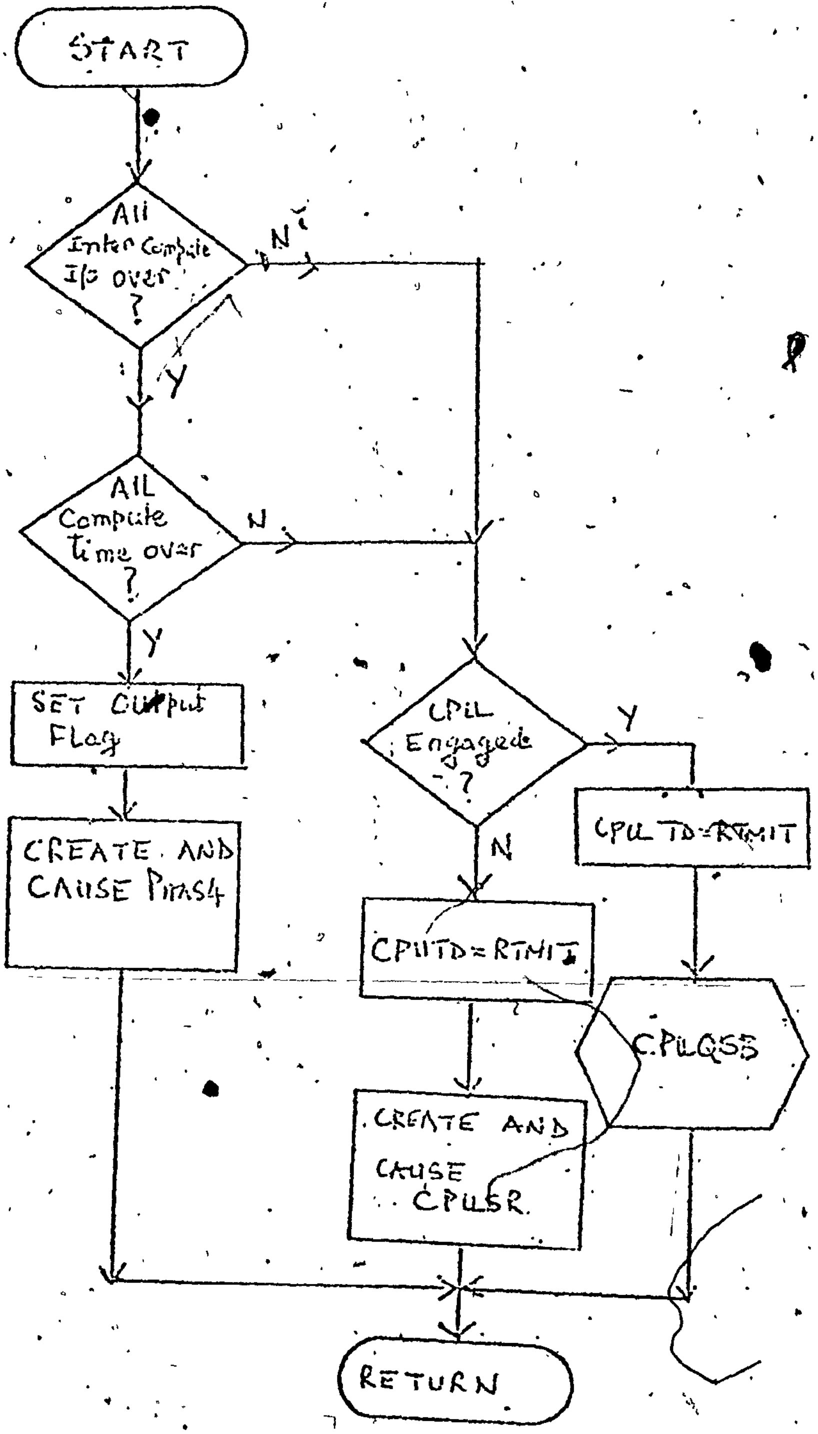
EXOGENOUS EVENT ANALYZ



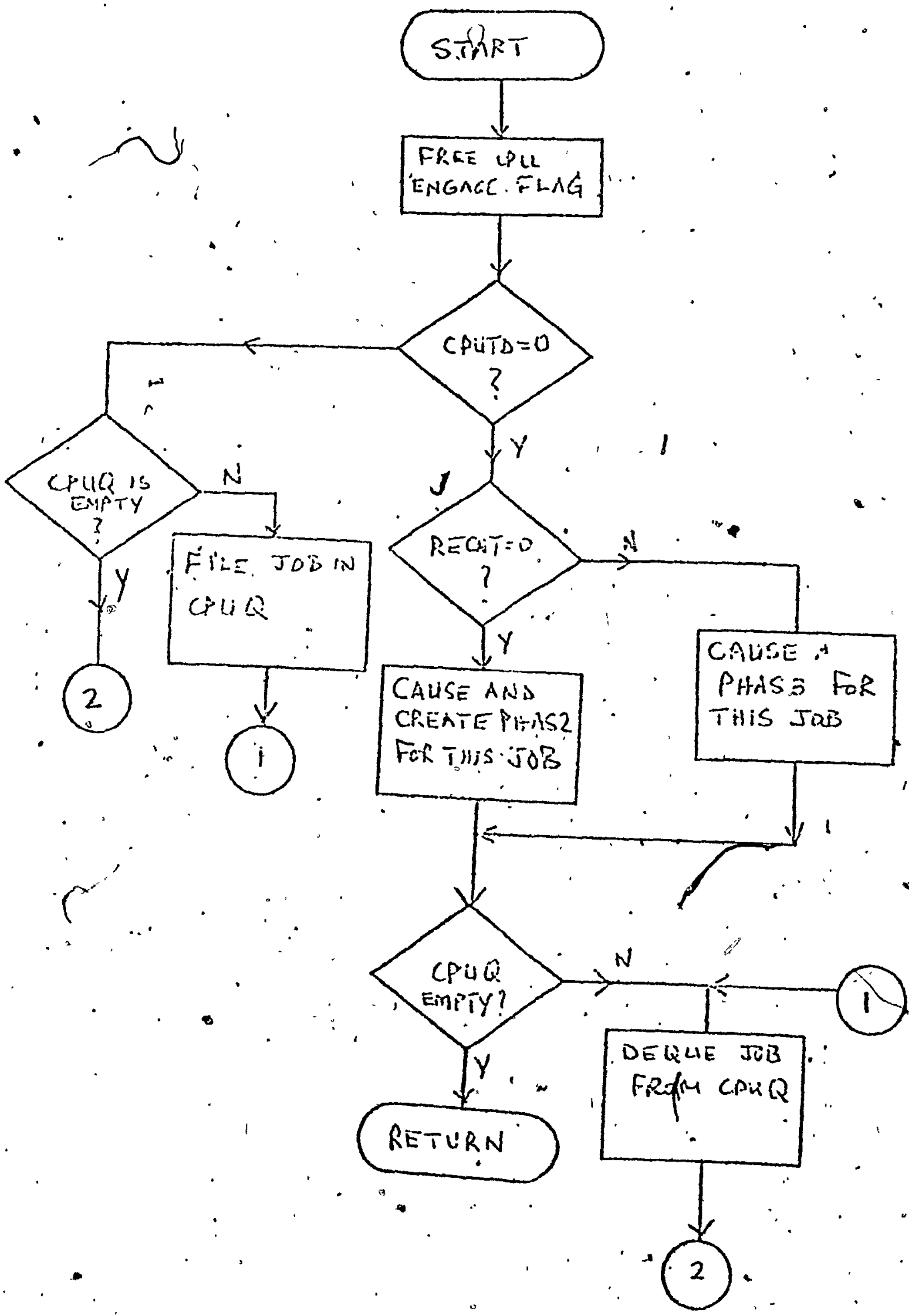
ENDOGENOUS EVENT PHAS1

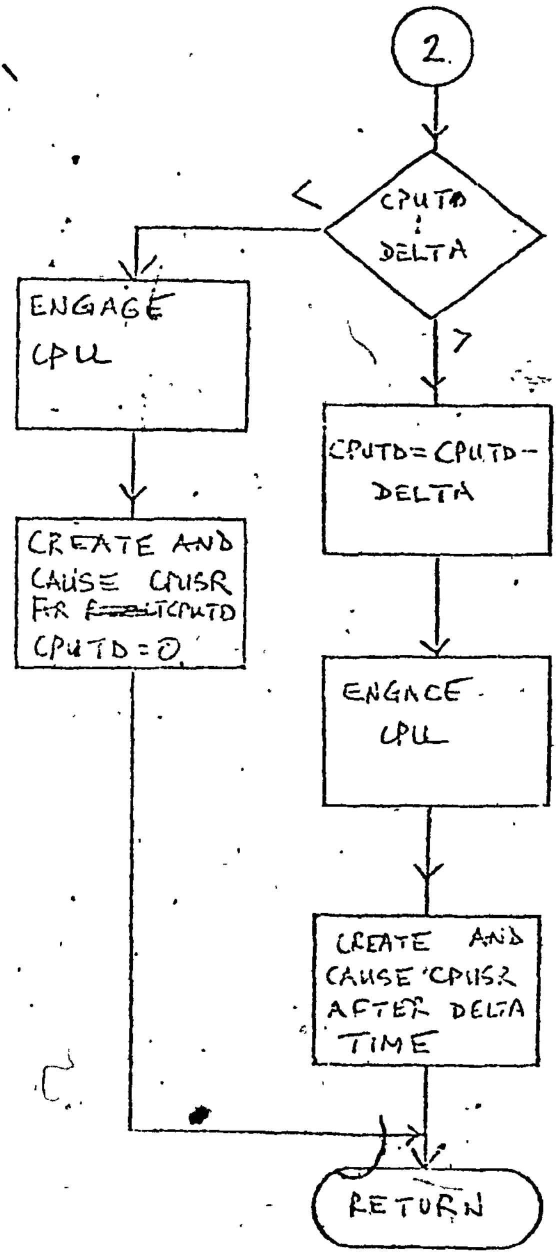

START
AVAILABLE MEMORY : MCH. DEMD
<
≥
CREATE AND CAUSE PHAS2
CMQSUB
GRANT CENTRAL MEMORY & UPDATE AVAILABLE MEMORY
RETURN

ENDOGENOUS EVENT PHASE

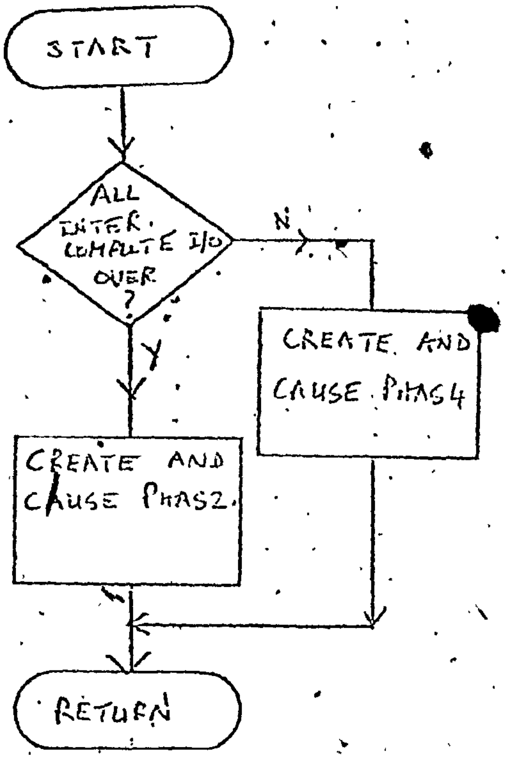


ENDOGENOUS EVENT CPUSR (CPU SCHEDULER)

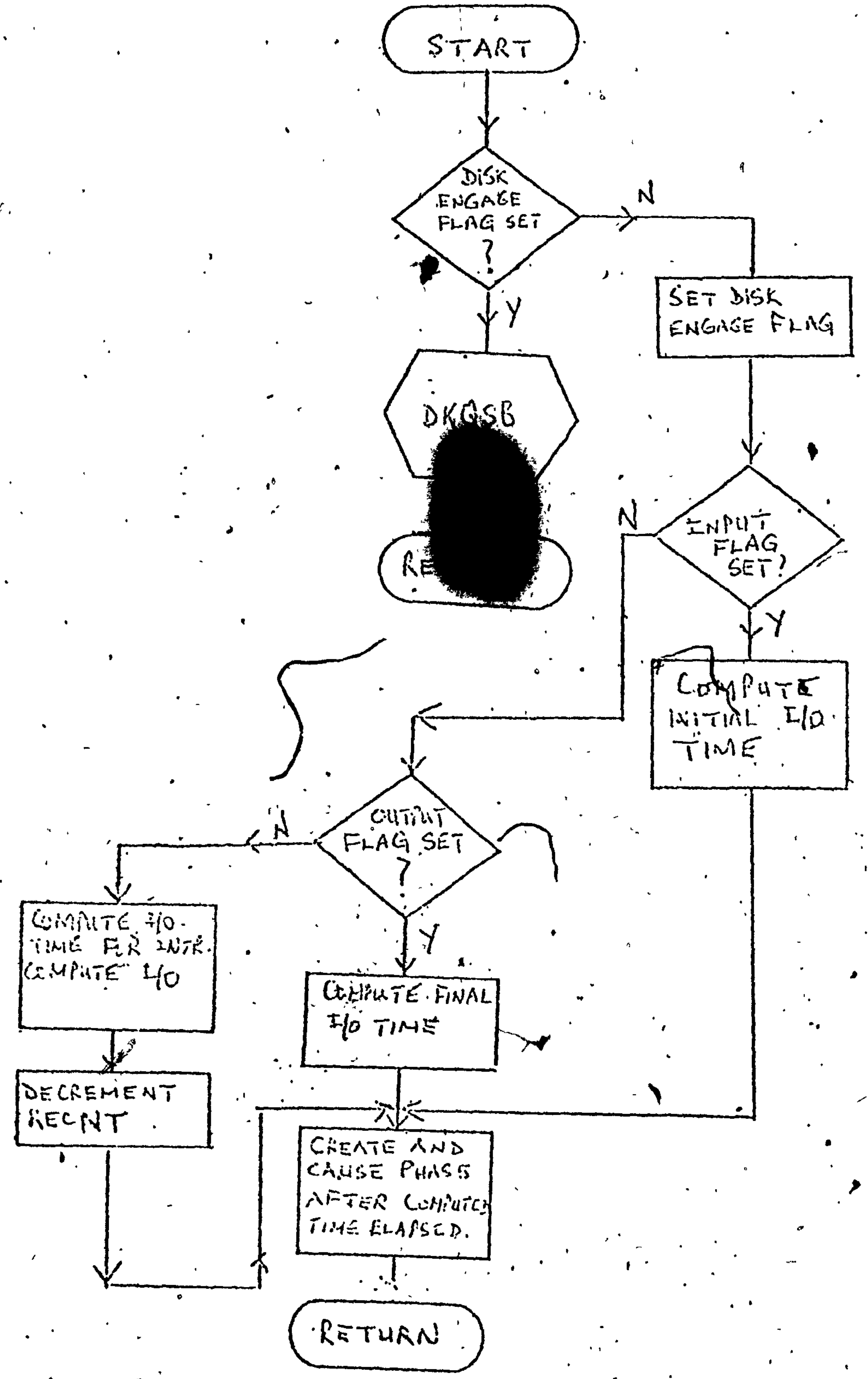



2.
<
CPUTD
:
DELTA
>
ENGAGE CPU
CREATE AND CAUSE CPUSR FOR CPUTD
CPUTD=0
CPUTD=CPUTD-DELTA
ENGAGE CPU
CREATE AND CAUSE CPUSR AFTER DELTA TIME
RETURN

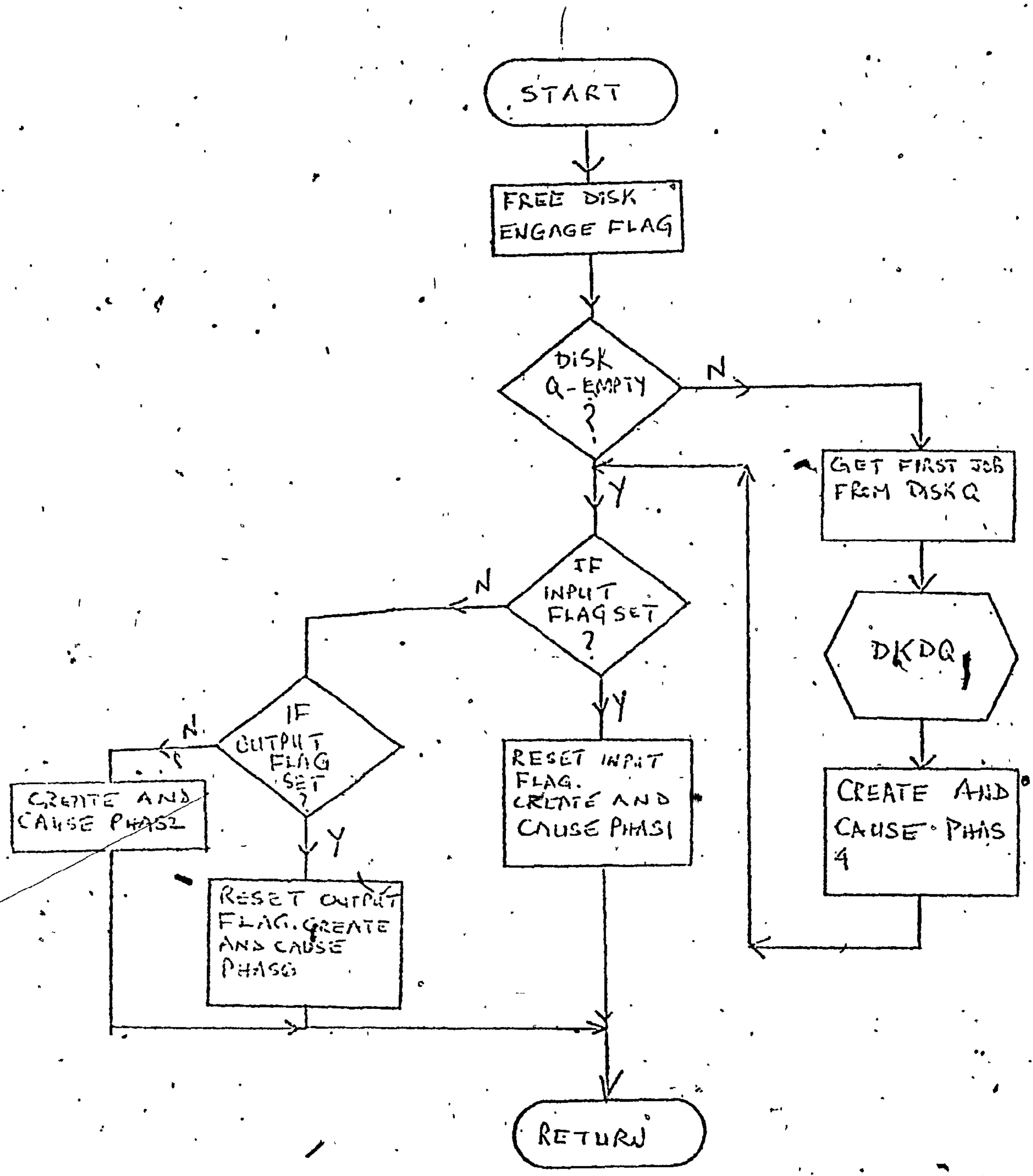
ENDOGENOUS EVENT PHAS 3



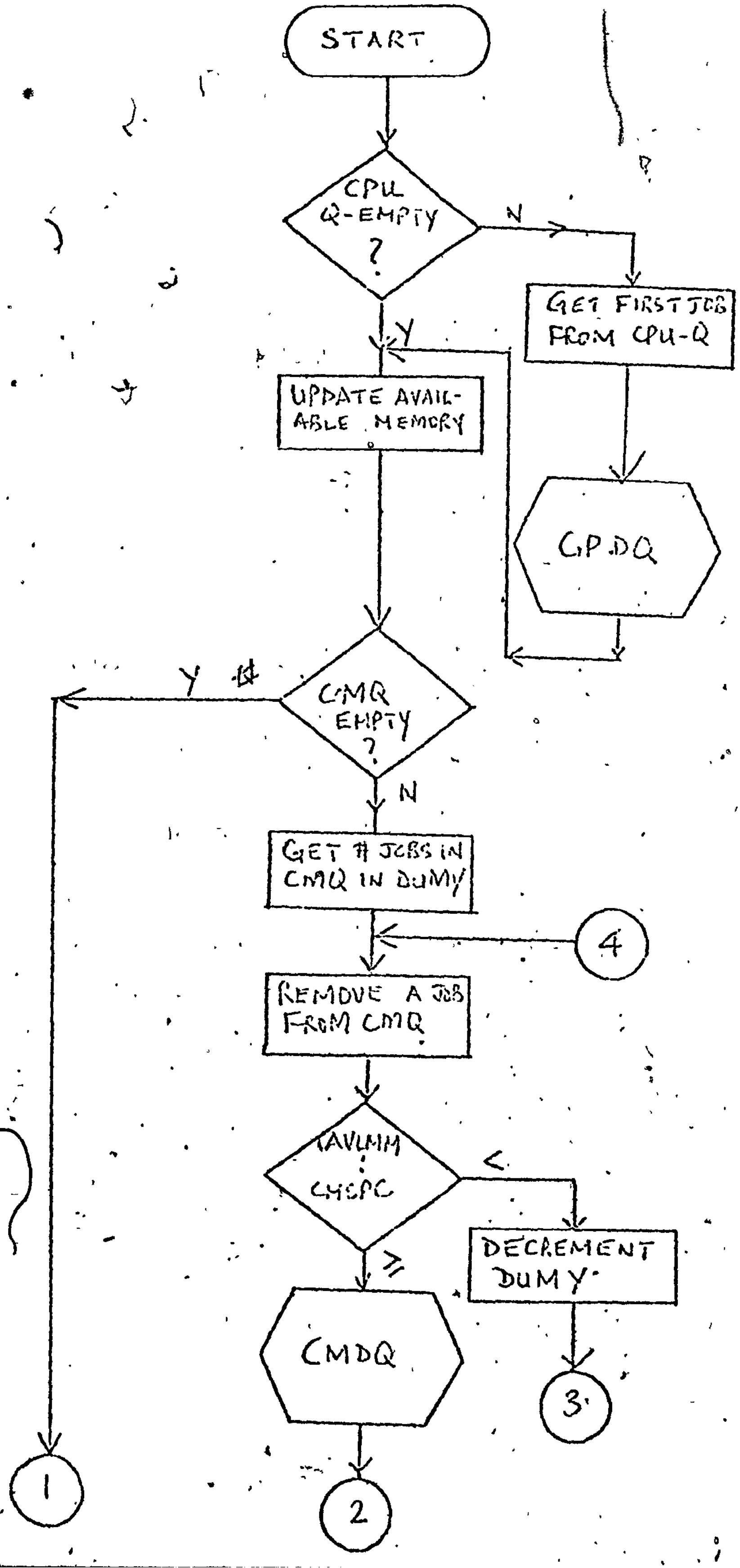
ENDOGENOUS EVENT PHAS4

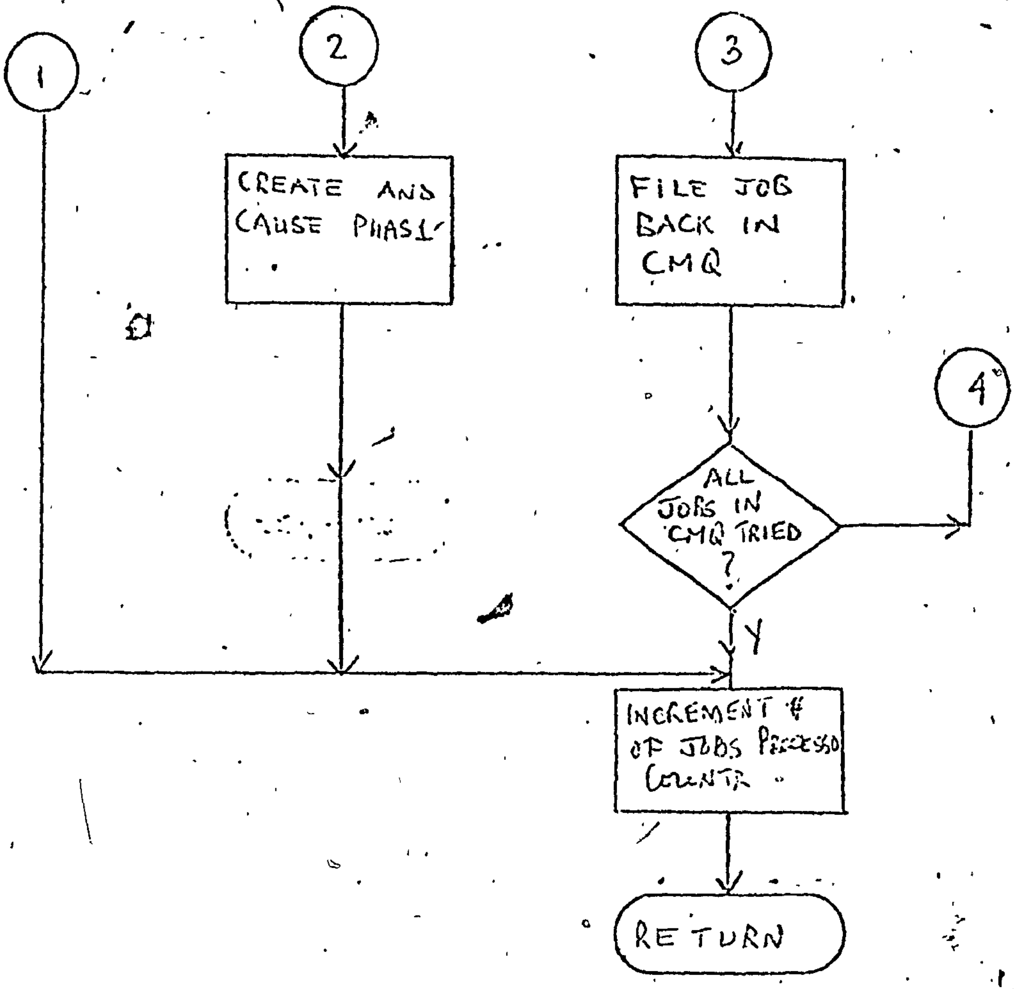


ENDOGENOUS EVENT PHAS 5

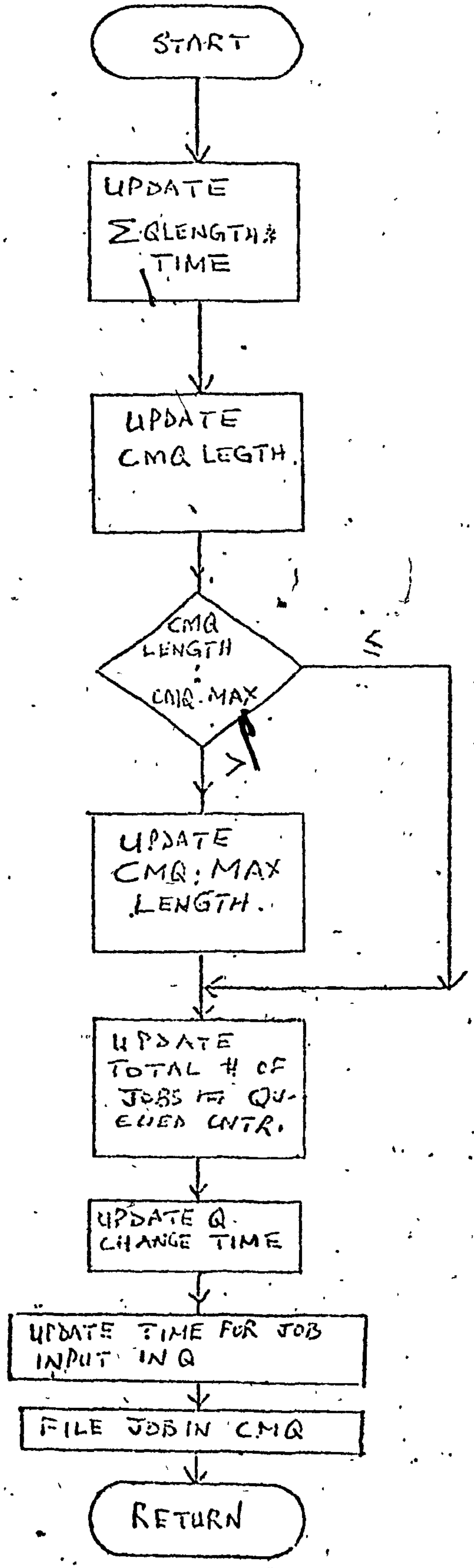


ENDOGENOUS EVENT PHASE

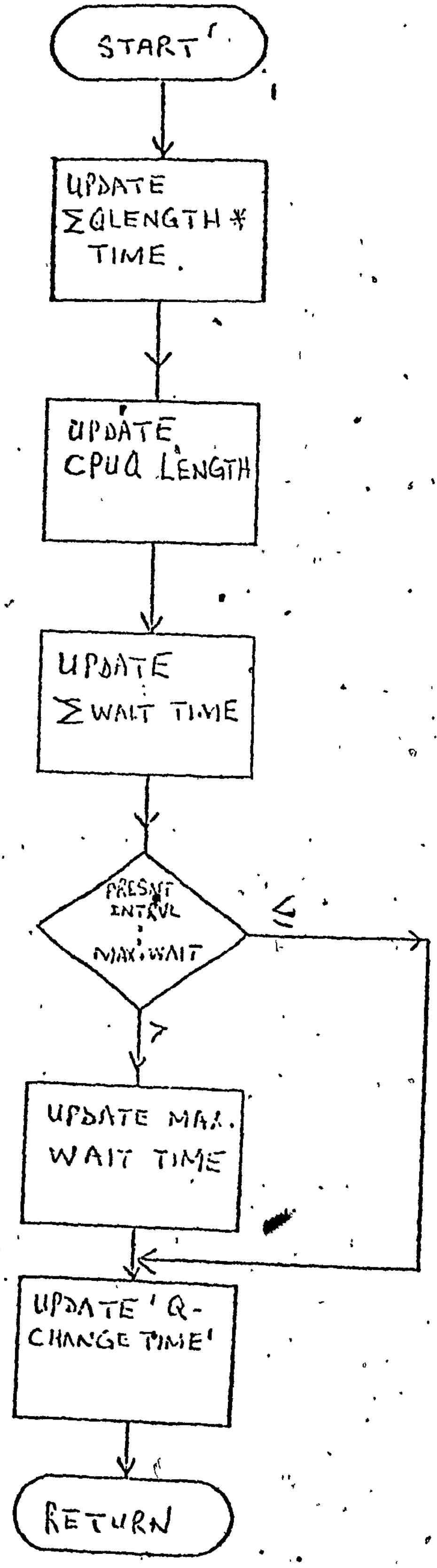



1
2
3
CREATE AND CAUSE PHAS1
FILE JOB BACK IN CMQ
4
ALL JOBS IN CMQ TRIED ?
Y
INCREMENT # OF JOBS PROCESSD COUNTR
RETURN

SUBROUTINE CMQSUB



SUBROUTINE CPDQ



1. Subroutine CPUQSB and DKQSB are similar to CMQSB.

2. Subroutine DKDQ and CMDQ are similar to CPDQ.

BIBLIOGRAPHY.

A1. V.A. Abell, et. al., Scheduling in a general purpose operating system, FJCC, 1970.

B1. L.A. Belady, A study of replacement algorithms for virtual storage computer, IBM. Syst. Journal, 5,2, 1966.

B2. U.N. Bhat, Elements of applied stochastic processes.

B3. J. Blatny, et. al., On optimization of performance of Time-sharing system by simulation, Comm. ACM., June 1972.

C1. W. Chang., and D.J. Wong., Analysis of Real-time Multiprogramming, J.ACM. 12, 4, Oct. 1965.

C2. P. Calingart, System performance evaluation, survay and appraisal, Comm. of ACM., 10, 1, Jan. 1967.

C3. E.F. Codd, Multiprogram Scheduling, Comm. of ACM., 1960.

C4. E.G. Coffman, Markov chain analysis of Multiprogrammed systems, Naval Res. Logist. Quart., 16, 2, June 1969.

C5. H.N. Cantrell, and A.L. Ellison, Multiprogramming system performance measurement and analysis, SJCC, 1968.

C6. Clarke, and Disney, Probablity and Random process for Engineers and Scientist.

C7. Coffman, and Muntz, Models of pure Time-sharing disciplines for resource allocation, Proc. ACM. 24th. Nat. Conf., 1969.

D1. W.M. DeMeis, and N. Weizer, Measurement and analysis of a Demand paging Time-sharing systems, Proc. ACM. 24th. Nat. Conf., 1969.

D2. P.J. Denning, The working set model for program behavior, Comm. of ACM., 11, 5, May 1968.

D3. E.W. Dijkstra, The structure of the T.H.E. Multiprogramming system, Comm. ACM., 11, 5, May 1968.

F1. H. Frank, Analysis and optimization of Disk storage devices for Time-sharing systems, J. ACM., 16,4 , Oct. 1969.

G1. R.M. Graham, Performance prediction, Advanced course on software engineering, Edited by F.L. Bauer, Springer-Verlag.

G2. D.P. Jr. Gaver, Probablity models for Multiprogramming computer systems, J. ACM., 14, 3, July 1967.

H1. A.N. Habermann, Prevention of system deadlocks, Comm. ACM., 12, 7; July 1969.

H2. J.N.P. Hume, and C.B. Rolfson, Scheduling for fast turnaround in Job-at-a-time processing, Information processing 68-North Holland publishing company-Amsterdam, 1969.

H3. H. Hellerman, and H.J. Smith, Throughput analysis of some idealised input, output and Compute Overlap configarations, Computing Surveys, Vol. 2, No. 2, June 1970.

J1. R.M. Jones, Factors affecting efficiency of Virtual memory, IEEE Trans. Computers, C-18, 11, Nov. 1969.

K1. J.H. Katz, An experimental model of System/360, Comm. ACM., 10,11, Nov. 1967.

K2. A.D. Karush, Two approaches for measuring the performance of Time-sharing systems, Software Age, April 1970.

M1. J.W. McGredie, Measurement criteria for Virtual memory paging rules, Proc. 24th. Nat. Conf. ACM. 1969.

M2. R.G. Miller, Priority queuees, The Annals of Mathematical statistics, Vol.31, 1960.

M3. M.H. MacDougal, Simulation of an ECS-based operating system, SJCC, 1967.

M4. Morse, Queues, Inventories and Maintenance.

N1. N.R. Nielson, An analysis of some Time-sharing techniques, Comm. ACM., 14,2, Feb. 1971.

O1. G. Oppenheimer, and N. Weizer, Resource management for a medium scale Time-sharing operating system, Comm. ACM., May 1968.

P1. T.B. Pinkerton, Performance monitoring in a Time-sharing systems, Comm. ACM., 12,11, Nov. 1969.

S1. D.F. Stevens, On overcoming High-priority paralysis in Multiprogramming system, A case history, Comm. ACM., August 1968.

S2. S. Sherman, et., al., Trace driven modelling and analysis of CPU scheduling in a Multiprogramming system, Comm. ACM., Dec. 1972.

W1. V.L. Wallace, and L. Mason, Degree of Multiprogramming in Page-on-demand systems, Comm. ACM., 12,6, June 1969.

W2. W.A. Wulf, Performance monitors for Multiprogramming systems, Proc. Sec. Symp. Oper. Syst. Principle, Oct. 1969.

Y1. A.C. Yeh, An application of statistical methodology in the study of computer system performance, Statistical Computer performance evaluation, Edited: W. Freiberger.