**DIGITAL SYSTEMS LABORATORY**

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY . STANFORD, CA 94305


SU–326–P.39–25 SEL 77–022

# PROPERTIES AND APPLICATIONS OF THE LEAST-RECENTLY-USED STACK MODEL

by
B. Ramakrishna Rau

**May 1977**

**Technical Report No. 139**

The work described herein was supported in part by the U.S. Energy Research and Development Administration under contract No. EY-76-S-03-0326-PA 39. Computer time was made available by the Stanford Linear Accelerator Center.

SEL 77-022

SU-326-P.39-25

# PROPERTIES AND APPLICATIONS OF THE LEAST-RECENTLY-USED STACK MODEL

by

B. Ramakrishna Rau

May 1977

Technical Report No. 139

DIGITAL SYSTEMS LABORATORY
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

The work described herein was supported in part by the U.S. Energy Research and Development Administration under contract No, EY-76-S-03-0326-PA 39.
Computer time was made available by the Stanford Linear Accelerator Center

SEL 77-022 SU-326-P.39-25

DIGITAL SYSTEMS LABORATORY

Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Technical Report No. 139

# PROPERTIES AND APPLICATIONS OF THE LEAST-RECENTLY-USED STACK MODEL

by

B. Ramakrishna Rau

May 1977

## ABSTRACT

The Least-Recently-Used Stack Model (LRUSM) is known to be a good model of temporal locality. Yet, little analysis of this model has been performed and documented. Certain properties of the LRUSM are developed here. In particular, the concept of the Stack Working Set is introduced and expressions are derived for the forward recurrence time to the next reference to a page, for the time that a page spends in a cache of a given size and for the time from last reference to the page being replaced. The fault stream out of a cache memory is modelled and it is shown how this can be used to partially analyze a multilevel memory hierarchy. In addition, the Set Associative Buffer is analyzed and a necessary and sufficient condition for the optimality of the LRU replacement algorithm is advanced.

Keywords and Keyphrases:
Models of program behavior, Analysis of program behavior, Least-Recently-Used Stack Model, Stack Working Set, Analysis of Multilevel Memory Hierarchies, Set Associative Buffer, Temporal Locality

CR Category Numbers:
6.34, 8.1

The work described herein was supported in part by the U.S. Energy Research and Development Administration under contract No. EY-76-S-03-0326-PA 39. Computer time was made available by the Stanford Linear Accelerator Center.

# 1 Introduction.

Program behavior impacts performance in numerous ways. Of particular interest to us here will be those aspects of program behavior which reflect upon the performance and the operation of multi-level memory hierarchies. Every replacement algorithm has the common feature that when a page is fetched to a particular level of the hierarchy, it is kept resident for a certain length of time before being discarded (or destaged to a lower level). The residency time is a function of, amongst other factors, the replacement algorithm. The larger the fraction of references to a page that occur during the page's residency period, the greater the success of the algorithm. Correspondingly, the program property which results in references to a particular page being clustered closely in time, contributes to the effectiveness of a memory hierarchy irrespective of the replacement algorithm that is employed. This clustering of references does in fact exist in most programs to some extent and is termed temporal locality. Locality, as defined by Denning is the tendency of programs to concentrate their references over a significant interval to a relatively small subset of their address space, and for this favored subset to gradually change [DENN68, DENN72]. It should be evident that the definition is equivalent to that of temporal locality. The favored subset is the collection of pages that are currently in the midst of a cluster of references.

The study of temporal locality may be approached either empirically or analytically. The analytic studies center around a model for temporal locality. Two conflicting demands are made on these models of program behavior; they must be analytically tractable to permit derivation of useful results and they must also portray, accurately, the properties of the program's reference string. The latter requirement is necessary if the derived results are to have any value since an elegant analysis based upon a faulty model is meaningless. However, an increase in the accuracy of the model generally involves a corresponding increase in its complexity, which eventually results in total intractability. Clearly, a compromise is needed to permit fairly accurate results by coupling a reasonably realistic model with a tractable analysis.

## 2 The Modelling of Temporal Locality

A survey of the literature reveals very few analytically tractable models of program behavior. The Independent Reference Model, (IRM), is, perhaps, the model which has been investigated and analyzed most thoroughly, [FRAN74, KING71, RAO75]. Its major drawback has been in its inability to capture the dynamic, short-term referencing behavior of programs which is closely tied to temporal locality. The IRM, which is based on the long term average reference probability of each page, is quite' inadequate in this respect. The AO-Inversion model, based on the IRM, rectifies this situation [RAFI76]. The Working Set Model, (WSM), and the Least-Recently-Used Stack Model, (LRUSM), are both able to model temporal locality successfully [DENN68, DENN72, COFF73, MATT70, SHEM66].

The former has the advantage of being the more natural model since it is based on the inter-reference interval statistics and, so, measures the clustering more directly. It also is able to predict the performance of the Working Set replacement algorithm exactly. The latter is more compact in the number of statistics, can be used for simulation purposes to generate a string of memory references which is consistent with the model and it predicts the performance of the LRU replacement algorithm exactly. The WSM consists of the probability function, F(t), where F(t) is the probability (averaged over all pages) that a reference to a page occurs exactly t references after the previous reference to that page. This model flows quite naturally from the definition of temporal locality. However, since the range of the inter-reference interval, t, is unbounded, we need, in principle, to measure and retain an infinite number of statistics. The LRUSM is obtained by maintaining the pages in a stack in increasing order of time since last reference. The statistics gathered are the average probability, P(d), of referencing the page which is in position d for all possible values of d. Since d cannot exceed the number of pages, the number of statistics to be gathered is limited by the size of the program. It is in this sense that the LRUSM is a compact model. P(d) may also be interpreted as the probability of referencing d distinct pages between two successive references to the same page, and though this would seem to be a somewhat artificial model of temporal locality it, nevertheless, captures most of the flavor of the WSM.

The LRUSM has the advantage in that it is possible to generate a reference string using the model. This is because the LRUSM, at each instant, defines a distribution over the pages describing their probability of reference. Having generated a reference string, we can perform any measurements we choose upon it to obtain results we could not have arrived at analytically. The WSM does not have this capability. Using this technique Rafii has shown that the LRUSM predicts the performance under the MIN replacement algorithm ([BELA66]) accurately and that it does a fair job of predicting the Working Set size and fault rate under the Working Set replacement algorithm [RAFI76]. These properties make the LRUSM very attractive. Yet, little analysis of the LRUSM has been performed and documented. In the subsequent sections we shall list some properties and applications of the LRUSM. Before that, we shall state the assumptions behind the LRUSM. Also, since extensive use is made of generating functions, a few of the more important properties of generating functions will be listed here.

## 3 The Least-Recently-Used Stack Model.

Let us assume that we are modelling a program which makes references to N distinct pages, The LRUSM orders these pages in a stack by the time since the last reference to them, with the most recently referenced page at the top of the stack. Let the stack positions be numbered so that the top of the stack is position 1 and the bottom of the stack is position N. Then, associated with position $i$ is the probability $p(i)$

that the next reference will be to the page which is currently in position i. It is assumed that

1. The probability of referencing the page in position i is independent of the identity of the page, and
2. The probability p(i) is time-invariant, and
3. The probability p(i) is independent of the past reference activity. (However, since the state of the stack is definitely dependent on the past reference behavior, the probability of referencing a particular page is also dependent on the past references).

The next reference is obtained by sampling from the distribution {p(i)}, and picking the page in that position. To maintain the LRU ordering, this page is moved to the top of the stack. The position i in which the page was found is termed the stack distance of that reference. The stack distance string is generated by a zero-th order Markov chain. This accounts for the tractability of the LRUSM. The page reference process, on the other hand, is far more complicated. In fact, it cannot be described by any finite order Markov chain. The individual page reference processes are renewal processes, but they are not independent. In particular, two pages cannot be referenced simultaneously.

The stack distance probability distribution is estimated by driving the LRU stack with the reference string of the program being modelled. For each reference , the stack is scanned to determine the position of the referenced page. A count associated with that position is incremented and the LRU stack is updated. These counts are normalized

with respect to the total number of references to obtain the fraction of the references which were to each position. These fractions are the estimate of {p(i)}.

The hit ratio, h(n), for a memory with a capacity of n pages under the LRU policy is given by

$$h(n) = \sum_{i=1}^{n} p(i)$$

The miss ratio, i.e., the probability that the referenced page is not in the memory is given by m(n) = 1-h(n). In the LRUSM, both m(n) and h(n) are time-invariant and independent of the identity of the pages in the memory. Consequently, the instants of page fault occurrence form a series of Bernoulli trials and the interval between two successive faults is geometrically distributed with a mean of 1/m(n). Furthermore, each such interval is independent of all other such intervals. Figure 1 displays the Markov chain for the path of a page through the LRU stack. It will be helpful to refer to it in the subsequent discussion.

## 4 Probability Generating Functions.

Consider a sequence of numbers, $r=\{r_0, r_1, \ldots\}$. The generating function for this sequence is given by

$$G(z) = \sum_{i=0}^{\infty} r_i z^i$$

The mapping to and from the generating function is unique. If the sequence were a probability distribution, $p=\{p_0,p_1,p_2,\ldots\}$ where $p_i$ is the probability of some integer valued random variable X assuming the value i, the corresponding generating function is termed the probability generating . function (p.g.f.). Perhaps the most important feature of a p.g.f. is that it permits convolution to be represented as a multiplication. Therefore, if G(z) and H(z) are the p.g.f.'s for two independent integer valued random variables X and Y respectively, then the p.g.f. for the sum X+Y is given by G(z)H(z). In addition the p.g.f. has the following properties:

$$G(z)\Big|_{z=1} = G(1) = \sum_{i=0}^{\infty} p_i = 1 \quad \text{if the distribution is honest,}$$

$$G'(z)\Big|_{z=1} = \sum_{i=0}^{\infty} i p_i z^{i-1}\Big|_{z=1} = \sum_{i=0}^{\infty} i p_i = \text{mean (X)}$$

$$G''(z)\Big|_{z=1} = \sum_{i=0}^{\infty} i(i-1) p_i z^{i-2}\Big|_{z=1} = \sum_{i=0}^{\infty} (i^2-i) p_i = \text{Var(X)}+[\text{Mean(X)}]^2-\text{Mean(X)}$$

Also, let S be the sum of N independent, identically distributed random variables $X_1,X_2,\ldots,X_N$ where N itself is a random variable, then if G(z) is the p.g.f. for $X_j$ and H(z) is the p.g.f. for N, the p.g.f. for S is given by H(G(z)). The reader interested in the derivations of these properties and in a further discussion of probability generating functions is referred to Feller's book [FELL68].

## 5 Properties of the LRUSM

One of the few properties of the LRUSM that has been documented is that every ordering of the pages in the LRU stack is equally probable. This follows directly from the fact that the probability transition matrix for the Markov chain describing the LRUSM is doubly stochastic. As a result, each page spends, on the average, an equal amount of time in each stack position and each page has an equal long-term probability of being referenced. We shall concentrate here on deriving the p.g.f. and/or the moments of the lengths of various intervals which are of interest.

### 5.1 The Stack Working Set

We shall often find it helpful to use the concept of the Stack Working Set. Let us imagine an LRU stack which is initially empty. The first reference will be a fault and the corresponding page is placed at the top of the empty stack. Each subsequent fault will cause the number of pages in the stack to be increased by 1. After some time, t, the number of pages in the stack will be some value **w.** w is the size of the Working Set with a window size of t [DENN68]. The average working set size for a window of size t will be given by W(t). Alternatively, we might be interested in the average time T(w) that it takes to build up a working set of size w. T(w) is a function that is defined only for integral values of **w.** Let us assume that it is extended by interpolation so as to be defined for all positive values and in a manner that ensures that it is continuous. Then it is possible to

define the inverse function D(t) such that $D(T(w)) = w$. The function D(t) is analogous to the function W(t) but is not, in general, identical. D(t) is of interest since it is easily obtained from the LRU stack model. Hence it is named the Stack Working Set (SWS). W(t) on the other hand, cannot be derived easily from the LRUSM.

THEOREM 1. The probability generating function (p.g.f.) for and the mean of the time required to build up a Stack Working Set of d pages are given by

$$\tau(z;d) = \prod_{i=0}^{d-1} m(i).z/(1-h(i)z) \qquad \text{and}$$

$$T(d) = \sum_{i=0}^{d-1} 1/m(i) \qquad \text{respectively.}$$

PROOF. Consider a SWS that has just attained the size d-l. Under the assumptions of the LRUSM, the time to the next fault is independent of the time taken to build up the SWS of size d-l. Then, since the p.g.f. of the time to the next fault is given by

$$\sum_{j-1}^{\infty} m(d-1)(h(d-1)^{j-1}z^j = m(d-1)z/(1-h(d-1)z),$$

we have

$$\tau(z;d) = \tau(z;d-1).m(d-1)z/(1-h(d-1)z)$$

Noting that $\tau(z;1) = z$, by induction we have that

$$\tau(z;d) = \prod_{i=0}^{d-1} m(i)z/(1-h(i)z)$$

Differentiating and putting $z=1$, we also have that

$$T(d) = \tau'(1;d) = \sum_{i=0}^{d-1} 1/m(i)$$

Q.E.D.

The SWS concept comes in handy in many places. We shall use it to give an informal proof for Theorem 2. It is the basis for characterizing the spatial locality of a reference string [RAU75]. We shall also suggest a procedure to determine the amount of memory that gets allocated to each of a number of concurrent reference streams in a shared memory which is managed by a global LRU replacement algorithm.

## 5.2 Analysis of Multilevel Memory Hierarchies.

A problem of increasing importance is the evaluation of a multilevel memory hierarchy. The emergence of technologies such as CCD and bubble memories makes multilevel hierarchies attractive. The analysis of such a structure is confronted by at least two significant obstacles:

1. The request stream that a particular level sees is the fault stream from the level immediately above it in the hierarchy. The model for the request stream to the highest level does not describe the request stream to any of the lower levels. A procedure is needed to characterize the fault stream in terms of the model for the request stream and the parameters that describe the structure of the level under consideration.

2. The unit of transfer (page size) between each pair of adjacent levels will, generally, be different. The fault stream out of one level

will be in terms of one page size, but the request stream to the lower level must be in terms of another (larger) page size. A method for transforming the page size of the model for a request stream is needed. This is shown by Rau to be related to the problem of suitably modelling the spatial locality of the request stream [RAU75]. The desired transformation is obtained there. We shall not consider this problem any further.,

Theorem 4 is a first attempt at tackling the question of modelling the fault stream. Theorems 2 and 3 prepare the ground by proving certain results needed in Theorem 4. We wish first to derive an expression for the mean replacement time, i.e., the time that it takes for a page to be replaced from a memory of capacity d pages measured from the time of last reference to that page.

THEOREM 2. In a memory of capacity d pages, under the LRU replacement algorithm, the p.g.f. for and the mean of the time to replace a particular page, measured from the time that it was last referenced are given by

$$\mathcal{V}(z;d) = \prod_{i=0}^{d-1} m(i)z/(1-h(i)z) \qquad \text{and}$$

$$V(d) = \sum_{i=0}^{d-1} 1/m(i) = T(d) \qquad \text{respectively} \qquad (1)$$

INFORMAL PROOF. Under the LRU replacement algorithm, the page is replaced when it reaches position d+l in the LRU stack. This means that

a SWS of size d (and not including the page in question) must be built up from the time that the page was last referenced and at the top of the LRU stack. Then, by Theorem 1, we have the desired results. However, the requirement that the page under consideration not be included in the SWS raises some doubts regarding the applicability of Theorem 1 in this situation. Accordingly, a more rigorous proof is advanced.

PROOF. Firstly, under the assumptions of the LRUSM, the time spent in each position is independent of the times spent at other positions. Accordingly,

$$v(z;d) = \prod_{i=1}^{d} u(z;i)$$

where $u(z;i)$ is the p.g.f. for the time that a page spends in position i given that the next distinct position is i+1 (and not 1). Now the probability that a page spends time j in position i given that the next position visited is i+l

$$= \frac{P[\text{spends time } j \text{ in position } i \text{ and next position is } i+1]}{P[\text{next position is } i+1]}$$

$$= \frac{(h(i-1))^{j-1}.m(i)}{P[\text{next position is } i+1]}$$

Therefore,

$$u(z;i) = \sum_{j=1}^{\infty} m(i)\,(h(i-1))^{j-1} z^j / P[\text{next position is } i+1]$$

$$\frac{m(i)z}{1-h(i-1)z} \cdot \frac{1}{P[\text{next position is } i+1]}$$

But P[next position is i-1] = $\sum_{j=1}^{\infty} m(i).(h(i-1))^{j-1} = m(i)/m(i-1)$

so $u(z;i) = m(i-1)z/(1-h(i-1)z)$

and so $v(z;d) = \prod_{i=0}^{d-1} m(i)z/(1-h(i)z)$

Differentiating and putting $z=1$ we have

$$V(d) = \sum_{i=0}^{d-1} 1/m(i)$$

Q.E.D.

The second result needed to prove Theorem 4 relates to the residency time of a page in a memory level of capacity d pages.

THEOREM 3. The mean residency time of a page in a memory level of capacity d pages under the LRU replacement algorithm, measured from the time that the page was staged up to that level to the time that it is next replaced is given by

$$W(d) = d/m(d) \qquad (2)$$

PROOF. The proof is facilitated by the use of probability generating functions to obtain a recurrence relation. Let $G(z;d)$ be the p.g.f. for the residency time in a memory of capacity d pages, i.e.,

$$G(z;d) = \sum_{i=0}^{\infty} a(i;d)z^i$$

where $a(i;d)$ is the probability of the page residing exactly i time units in the memory of capacity d. Then clearly

$$G(z;d)\,|_{z=1} = \sum_{i=0}^{\infty} a(i;d) = 1 \quad \text{and}$$

$$\frac{\partial}{\partial z}\, G(z;d)\,|_{z=1} = \sum_{i=0}^{\infty} i \cdot a(i;d) = W(d)$$

Consider, now, the path of a page from the time that it is placed at the top of the stack to the time that it first enters position d+l in the stack. We can divide this interval into a number of sub-intervals X(i). The random variable X(1) is the time that it takes the page to first drop to position d plus the time that it spends in position d. The page can exit position d either by moving to the top of the stack with probability p(d) or by dropping to position d+l with probability m(d). If it moves to the top of the stack it goes through another sub-interval of length X(2). The p.g.f. N(y) for the number of sub-intervals corresponding to a residency period is given by

$$N(y) = \frac{m(d)}{m(d-1)}\, y + \frac{p(d)m(d)}{(m(d-1))^2}\, y^2 + \cdots$$

$$= \frac{m(d)}{m(d-1)}\, y \sum_{i=0}^{\infty} \left( \frac{p(d)}{m(d-1)}\, y \right)^i$$

$$= \frac{m(d)y}{m(d-1)-p(d)y}$$

since $m(d-1) = m(d) + p(d)$. The residency period is the sum of a random number of independent identically distributed random variables X(i).

Therefore, if H(z;d) is the p.g.f.for X

$$G(z;d) = N(H(z;d)) = \frac{m(d)\,H(z;d)}{m(d-1)-p(d)H(z;d)}$$

Now, under the LRUSM assumptions, X is the sum of two independent random variables -- the residency period in the top d-l positions of the stack plus the time spent in position d. Therefore,

$$H(z;d) = G(z;d-1) \cdot \frac{m(d-1)z}{1-h(d-1)z}$$

where the second term is the p.g.f. for the time spent in position d.

Now,

$$W(d) = G'(z;d)\big|_{z=1}$$

$$= m(d).\ \frac{H(z;d)\ p(d)H'(z;d)+H'(z;d)(m(d-1)-p(d)H(z;d))}{(m(d-1) - p(d)H(z;d))^2}\Bigg|_{z=1}$$

$$= \frac{m(d)m(d-1)H'(z;d)}{(m(d))^2}\Bigg|_{z=1} = \frac{m(d-1).H'(z;d)}{m(d)}\Bigg|_{z=1} \qquad (3)$$

And,

$$H'(z;d) = G(z;d-1).\frac{m(d-1)}{(1-h(d-1)z)^2} + G'(z;d-1).\frac{m(d-1)z}{1-h(d-1)z}$$

therefore,

$$H'(z;d)\big|_{z=1} = 1/m(d-1) + W(d-1) \qquad (4)$$

From Eqns. 3 and 4

$$W(d) = \frac{1}{m(d)} + \frac{m(d-1)}{m(d)} W(d-1)$$

Since W(1) = l/m(l), it is easily shown that d/m(d) satisfies the recurrence relation.

Q.E.D.

We can now return to the issue of modelling the fault stream out of a particular level in a memory hierarchy. Let the request stream to the k-th level in the hierarchy be characterized by the miss ratio function, $m_k(d_1)$, which gives the fault rate out of level k under the LRU replacement policy, if the capacity of that level is $d_1$ pages and where the fault rate is measured in units of faults out of level k per request to level k. For any given value of $d_1$, we wish to similarly characterize the fault stream (the request stream to level k+l) by a miss ratio function, $m_{k+1}(d_2)$, which gives us the fault rate out of level k+l as a function of the capacity of level k+l. We assume that the page size at levels k and k+l are the same.

THEOREM 4. Under the assumption that all pages have identical behavior, and if $W_k(n)$ and $V_k(n)$ are treated as exact rather than as average residency and replacement times, then if the request stream to level k is characterized by $m_k(d)$, and the size of the level is n pages, and if

the fault stream is characterized by the function $m_{k+1}(d)$ which gives the miss rate from level k+l (measured in faults at level k+l per request to level k) as a function of the capacity, d, of level k+l, then

$$m_{k+1}(d) = \begin{cases} m_k(n) & \text{for } d \leq n \\ m_k(d) & \text{for } d > n \end{cases}$$

PROOF: Let,

$$Q(t) = \frac{\partial D(t)}{\partial t} \quad \text{and}$$

$$F(t) = -\frac{\partial Q(t)}{\partial t}$$

where D(t) is the mean size of the Stack Working Set built up in time t.

By analogy with the Working Set Model, F(t) is the the probability that the length of the interval between two successive references to any given page is t [DENN72]. Q(t) is the fault rate corresponding to the size of the Stack Working Set built up over a period t. Note that by the definition of D(t) and T(n)

$$Q(t) = m(D(t)) \quad \text{and} \tag{5}$$

$$m(n) = Q(T(n)) \tag{6}$$

where T(.) is the inverse of the function D(.).

Let $F_k(t)$ and $Q_k(t)$ correspond to the request stream to level k, and $F_{k+1}(t)$ and $Q_{k+1}(t)$ correspond to the fault stream. For notational convenience, $W_k(n)$ and $V_k(n)$ will often be referred to as Wk and $V_k$, without introducing any ambiguity.

A page cannot be referenced at level k+l if a copy is present at level k, for the reference will be intercepted by level k. 'The period of residency at level k measured from the time it was last referenced at level k+l (and staged up to level k) is $W_k(n)$. Therefore, neglecting the variance of the residency time, for $t \leq W_k(n)$,

$$F_{k+1}(t) = 0$$

If $t > W_k(n)$ then the page is no longer present in level k. By Theorem 2, the page was referenced $V_k(n)$ time units before the time that it was displaced from level k (Fig.2) if the variance of the replacement time is neglected. Therefore, for $t > W_k(n)$,

$$F_{k+1}(t) = F_k(t - W_k + V_k)$$

Therefore, we have

$$F_{k+1}(t) = \begin{cases} 0 & \text{for } t \leq W_k \\ F_k(t - W_k + V_k) & \text{for } t > W_k \end{cases}$$

Since, by definition, $F_{k+1}(t) = -\partial Q_{k+1}(t)/\partial t$, we have

$$Q_{k+1}(t) = Q_{k+1}(0) - \int_0^t F_{k+1}(x)\,dx$$

$$= \begin{cases} Q_{k+1}(0) - \int_0^t 0\,dx = Q_{k+1}(0) & \text{for } t \leq W_k, \\ Q_{k+1}(W_k) - \int_{W_k}^t F_k(x - W_k + V_k)\,dx & \text{for } t > W_k \end{cases} \qquad (7)$$

By the first part of this equation, $Q_{k+1}(W_k) = Q_{k+1}(0)$. If the working set size at level k+1 is 0, then every fault at level k is also a fault at level k+1, and so,

$$Q_{k+1}(0) = M_k(n)$$

But by Eqn.6,

$$M_k(n) = Q_k(T_k(n))$$

And since, by Eqn.1,

$$V_k(n) = T_k(n)$$

we have

$$Q_{k+1}(0) = Q_{k+1}(W_k) = Q_k(V_k) \tag{8}$$

Substituting $y = x - W_k + V_k$ in Eqn.7 and using Eqn.8 we get,

$$Q_{k+1}(t) = \begin{cases} Q_k(V_k) & \text{for } t \leq W_k, \\ Q_k(V_k) - \int_{V_k}^{t-W_k+V_k} F_k(y)\,dy & \text{for } t > W_k. \end{cases}$$

Therefore,

$$Q_{k+1}(t) = \begin{cases} Q_k(V_k) & t \leq W_k, \\ Q_k(t - W_k + V_k) & t > W_k. \end{cases} \tag{9}$$

Next, since by definition $Q(t) = \partial D(t)/\partial t$ we have

$$D_{k+1}(t) = \int_0^t Q_{k+1}(x)\,dx$$

$$= \begin{cases} \int_0^t Q_k(V_k)\,dx & t \leq W_k, \\ D_{k+1}(W_k) + \int_{W_k}^t Q_k(x - W_k + V_k)\,dx & t > W_k \end{cases}$$

And, since $D_{k+1}(0) = 0$,

$$D_{k+1}(t) = \begin{cases} Q_k(V_k) * t & t \leq W_k, \\ Q_k(V_k) * W_k + D_k(t - W_k + V_k) - D_k(V_k) & t > W_k \end{cases}$$

But since by Eqns. 1, 2, 6,

$$D_k(V_k) = D_k(T_k(n)) = n, \text{ and}$$

$$W_k(n) = n/m_k(n), \text{ and}$$

$$Q_k(V_k) = Q_k(T_k(n)) = M_k(n),$$

we have,

$$D_{k+1}(t) = \begin{cases} m_k(n) * t & t \leq W_k \\ D_k(t - W_k + V_k) & t > W_k. \end{cases} \quad (10)$$

Therefore, for $t \leq W_k$ we have

$D_{k+1}(t) \leq n$, by Eqn. 10 and

$m_{k+1}(D_{k+1}(t)) = Q_{k+1}(t)$ by Eqn. 5, and

$Q_{k+1}(t) = Q_k(V_k)$, by Eqn. 9, and

$Q_k(V_k) = Q_k(T_k(n))$, by Eqn. 1, and

$Q_k(T_k(n)) = m_k(n)$ by Eqn. 6.

Therefore,

$$m_{k+1}(D_{k+1}(t)) = m_k(n)$$

i.e. $m_{k+1}(d) = m_k(n)$ for $d \leq n$ (11)

For $t > W_k$ we have

$D_{k+1}(t) > n$, and

$Q_{k+1}(t) = m_{k+1}(D_{k+1}(t))$ by Eqn. 5, and since

$Q_{k+1}(t) = Q_k(t - W_k + V_k)$ by Eqn. 9, and

$D_{k+1}(t) = D_k(t - W_k + V_k)$ by Eqn. 10.

Therefore, by Eqn. 5,

$$Q_k(t - W_k + V_k) = m_{k+1}(D_k(t - W_k + V_k))$$

But, by Eqn.5,

$$Q_k(t - W_k + V_k) = m_k(D_k(t - W_k + V_k))$$

Therefore,

$$m_{k+1}(d) = m_k(d) \qquad \text{for } d > n \tag{12}$$

From Eqns. 11 and 12 we obtain the theorem in its final form:

$$m_{k+1}(d) = \begin{cases} m_k(n) & d \leq n, \\ m_k(d) & 1 > n. \end{cases}$$

Q.E.D.

Theorem 4 states that, for a constant page size, the top n pages in the LRU stack for level k+l will be in the higher level (of size n pages). Conversely, it also states that a page which is at a position lower than n in the LRU stack for level k+l will not be in level k. This is, admittedly, a rather simplistic analysis since it uses only the average residency and replacement times and does not take into account the variance. Consequently, validation of the analysis was sought through the use of trace driven simulation. The trace tapes used were

043 - Fortran execution,

049 - Cobol execution,

050 - Cobol compilation.

The fault rate function predicted by Theorem 4 is compared with the measured fault rate function in Figs. 3-5. The correspondence is quite good for the page sizes normally used in cache memories, but is poor for page sizes of about 1K bytes when the value of d is close to n.

## 5.3 Moments of the Inter-Reference Interval

In the LRUSM, the sequence of references to a page forms a renewal process. This is obvious, since a reference to a page moves it to the top of the stack and resets its state. Its future behavior and path through the LRU stack is independent of the past and statistically identical each time it is referenced and brought to the top of the stack. We now seek a description of this renewal process. We shall do so by deriving the probability generating function for the inter-reference interval in a recursive form. This permits us to obtain closed form expressions for the first two central moments of the inter-reference interval. In fact, we can simultaneously obtain the first two central moments for the forward recurrence time of a page which is in any position in the LRU stack.

THEOREM 5. In the LRUSM, the mean, Ml(i), and the variance, $M_2(i)$, of the forward recurrence time (to next reference) of the page in position i of the LRU stack are given by

$$M_1(i) = (N-i+1)/m(i-1) \qquad \text{and}$$

$$M_2(i) = \left[ 2 \sum_{j=i-1}^{N-1} (N-j)/m(i) - (N-i+1) - (N-i+1)^2/m(i-1) \right] /m(i-1)$$

respectively, where N is the total number of pages. In particular, the mean and the variance of the inter-reference time to a page are given by

$$M_1(1) = N \qquad \text{and}$$

$$M_2(1) = 2 \sum_{j=0}^{N-1} (N-j)/m(j) - N - N^2 \qquad \text{respectively.}$$

PROOF. Consider a page which has just reached position i in the LRU stack. Let the p.g.f. for the time to next reference be $T_i(z)$. One of two things can happen to the page; it spends a certain period of time at position i and is then either referenced with unconditional probability p(i) or displaced to position i+1 with unconditional probability m(i). However, given that the page leaves position i, the probabilities of these two outcomes are p(i)/m(i-1) and m(i)/m(i-1) respectively, since p(i)+m(i) = m(i-1). If the page moves to position i+1, the remaining time to reference is independent of the time spent in position i. Then, noting that the p.g.f. for the time spent in position i is given by m(i-1)z/(1-h(i-1)z), we have that

$$T_i(z) = \frac{m(i-1)z}{1-h(i-1)z} \cdot \frac{p(i)}{m(i-1)} + \frac{m(i-1)z}{1-h(i-1)z} \cdot T_{i+1}(z) \cdot \frac{m(i)}{m(i-1)}$$

$$= \frac{z}{1-h(i-1)z} \left[ p(i) + m(i)\, T_{i+1}(z) \right]$$

Noting that $T_N(z) = m(N-1)z/(1-h(N-1)z)$, we can obtain the p.g.f. for the forward recurrence time, $T_i(z)$, and the p.g.f. for the inter-reference time, $T_1(z)$, from the recurrence relation.

Using the recurrence relation, we can also obtain the mean and the variance of the forward recurrence time in closed form. We have that

$$T_i'(z) = \frac{\partial}{\partial z} \left[ \frac{z}{1-h(i-1)z} \right] \left[ p(i)+m(i)T_{i+1}(z) \right]$$

$$= [p(i)+m(i)T_{i+1}(z)] \frac{\partial}{\partial z} \frac{z}{1-h(i-1)z} + \frac{z}{1-h(i-1)z} \cdot m(i)T_{i+1}'(z)$$

Using the fact that $T_{i+1}(1)=1$, we have that

$$M_1(i) = T_i'(1) = \frac{1}{m_{i-1}} + \frac{m_i}{m_{i-1}} \cdot T_{i+1}'(1)$$

Since TN'(l) = l/m(N-l), it can be shown by induction that

$$M_1(1) = (N-i+1)/m(i-1)$$

Similarly,

$$T_i''(z) = [p(i)+m(i)T_{i+1}(z)] \cdot \frac{\partial^2}{\partial z^2} \frac{z}{1-h(i-1)z}$$

$$+ 2 \cdot m(i)T_{i+1}'(z) \cdot \frac{\partial}{\partial z} \frac{z}{1-h(i-1)z}$$

$$+ \frac{z}{1-h(i-1)z} \cdot m(i)T_{i+1}''(z)$$

from which, noting that $T_{i+1}(1) = 1$ and $T_{i+1}'(1)=(N-i)/m(i)$ we get that

$$T_i'(1)=2(h(i-1)+N-i)/m(i-1)^2 + m(i)T_{i+1}''(1)/m(i-1)$$

Then, since $T_N''(1)=2h(N-1)/m(N-1)^2$, it can be shown by induction that

$$T_i''(1) = \left[2\sum_{j=i-1}^{N-1}(N-j)/m(j) - 2(N-i+1)\right]/m(i-1)$$

But $M_2(i) = T''_i(1)+M_1-M_1(i)^2$. Therefore,

$$M_2(i)=\left[2\sum_{j=i-1}^{N-1}(N-j)/m(j)-(N-i+1)-(N-i+1)^2/m(i-1)\right]/m(i-1)$$

and, in particular, the variance of the inter-reference time is given by

$$M_2(1) = 2\sum_{j=0}^{N-1}(N-j)/m(j) - N - N^2$$

Q.E.D.

The forward recurrence time was measured from the time that a page entered position i of the LRU stack. By the memoryless property of the geometrically distributed residence of the page in position i, the forward recurrence time measured from some arbitrary time given that the page is in position i will have the same distribution.

## 5.4 Optimality of the LRU replacement policy

The criterion for the optimality of a realizable demand paging policy is that it replace the page with the longest expected time to next reference. Therefore, if the LRU policy is to be optimal, we must have that

$$M_1(i) \leq M_1(i+1) \qquad \text{for } 1 \leq i < N, \text{ i.e., that}$$

$$\frac{N-i+1}{m(i-1)} \leq \frac{N-i}{m(i)}$$

which reduces to

$$p(i) \geq m(i)/(N-i) = [1/(N-i)] \cdot \sum_{j=i+1}^{N} p(i)$$

i.e., each stack distance probability must be at least as large as the average of the probabilities of the greater stack distances. Coffman and Denning have stated that a sufficient condition for the LRU policy to be optimal for all buffer sizes is that $p(i) \geq p(i+1)$ for $1 \leq i < N$ [COFF73]. We have found a necessary and sufficient condition for optimality. That their condition implies ours is clear, for if $p(i) > p(i+1)$ for $1 \leq i < N$, then

$$p(i) \geq [1/(N-i)] \cdot \sum_{j=i+1}^{N} p(j) = m(i)/(N-i)$$

On the other hand, the converse is not always true. This is best demonstrated by an example. Let N = 5 and let the stack distance probabilities be (0.6, 0.1, 0.2, 0.05, 0.05). The expected forward recurrence times for each stack position are (5, 10, 10, 20, 20). The LRU policy is optimal in this case despite the fact that the stack distance probabilities are not monotonically non-increasing.

## 6 Applications of the LRUSM.

The preceeding Theorems have described some of the properties of the LRUSM. We shall now demonstrate a few applications of the LRUSM in analyzing memory management strategies. In particular, we shall analyze the performance of a set associative LRU policy and an optimal demand pre-paging algorithm. In addition, we shall consider a memory shared by a number of concurrently active programs and managed by the global LRU policy. We shall suggest a procedure for calculating the manner in which the memory is allocated amongst the active programs.

### 6.1 The Set Associative Buffer.

The set associative buffer is almost invariably used at the cache level to effect a compromise between the performance of the cache and the complexity of managing it [KAPL73]. Each block (page) in the main memory is mapped into a quotient class based on the lower order bits of the block address. In a set associative buffer of degree of associativity = a, each quotient class is associated with a unique set of a block frames in the buffer. Any block from a particular quotient

class may only be placed in one of the frames in the corresponding set. The decision as to which of the a frames is to be chosen is based on the LRU policy. We have, therefore, a fully associative buffer of a blocks per quotient class. When a=1 we have what is termed the direct mapping buffer, and when a is the number of frames in the buffer, we have the fully associative buffer. Our analysis is based upon the following assumptions:

1. The standard LRUSM assumptions hold.
2. The address of the block in position i in the LRU stack is independent of the address of the block in position j for all i,j and $i \neq j$.
3. The block in position i of the LRU stack is equally likely to belong to any of the a quotient classes, for all i.

Let the set associative buffer have q quotient classes (columns) and a degree of associativity of a (i.e. a rows). l/q is the probability that the block in position i of the LRU stack is in any particular one of the q quotient classes.

Define $m_s(q,a)$ to be the miss rate in the set associative buffer described above. Also, define $p_s(q,a)$ to be the probability that the referenced block is found at position a in the LRU stack corresponding to some quotient class. Then

$$m_s(q,a) = 1 - \sum_{j=1}^{a} p_s(q,j)$$

Now, $p_s(q,j)$ is the probability that exactly j-l of the blocks, which are above the referenced block in the fully associative LRU stack, fall into the same quotient class as the referenced block. Given that the referenced block is in position i of the fully associative LRU stack, the probability of there being j-l such blocks is given by $C(i-1,j-1).(1/q)^{j-1}.(1-1/q)^{i-j}$, where C(i-l,j-1) is the binomial coefficient $(i-1)!/[(j-1)!(i-j)!]$. Therefore, the probability of referencing a block and finding it at position j in the quotient class LRU stack is given by

$$p_s(q,j) = \sum_{i=1}^{N} p(i)\ C(i-1,j-1)(1/q)^{j-1}\ (1-1/q)^{i-j} \text{ and}$$

$$m_s(q,a) = 1 - \sum_{j=1}^{a}\sum_{i=1}^{N} p(i)C(i-1,j-1)(1/q)^{j-1}(1-1/q)^{i-j}$$

$$= 1 - \sum_{i=1}^{N} p(i) \sum_{j=1}^{a} C(i-1,j-1)(1/q)^{j-1}(1-1/q)^{i-j}$$

$$= 1 - \sum_{i=1}^{N} p(i)\ B_a(i)$$

where $B_a(i) = \sum_{j=1}^{a} C(i-1,j-1)(1/q)^{j-1}\ (1-1/q)^{i-j}$

and is a function of i, which is equal to 1 for $i \leq a$ and then monotonically decreases. The rate at which $B_a(i)$ decreases for $i>a$ is greater for larger values of a, assuming that the product aq is kept constant, i.e. the total capacity of the buffer is kept constant (Fig.6). Thus, the hit rate to a set associative buffer with a high degree of associativity gives heavier weighting to the p(i)'s for small

i and lower weighting to the p(i)'s for large i than does a set associative buffer of identical capacity but with a smaller degree of associativity. Letting c=aq, the extreme cases are:

the Direct Mapping Buffer

$$m_s(c,1) = 1 - \sum_{i=1}^{N} p(i)\,(1-1/c)^{i-1}$$

and the Fully Associative Buffer

$$m_s(1,c) = 1 - \sum_{i=1}^{c} p(i)$$

When the p(i) is monotonically decreasing, the miss ratio is minimized by weighting as heavily as possible the p(i)'s for small i. The fully associative buffer does just this. However, the fully associative buffer places no weight at all for $i>c$. Thus, if p(i) is of the form shown in Fig.7, the direct mapping buffer would perform better. In fact, just such an anomalous result was obtained when studying the trace tape 050. Subsequent measurement and examination of p(i) for the tape revealed a bell shaped function as in Fig.7.

## 6.2 Optimal Demand Pre-Paging

The principle of optimality, for a fixed memory allocation policy, states that the pages, for which references are most imminent, should be maintained in the buffer. If limited to demand paging policies, the optimal policy is MIN, which, on the occurrence of a fault, replaces the page in the buffer with the least imminent reference. If we permit

pre-paging, then the optimal policy, assuming we have n page frames available, is to always maintain in the buffer the n pages with the most imminent references. Such a policy would cause no page faults at all even for n=1. But such a policy is clearly impractical even if we had complete future knowledge. A more feasible pre-paging policy is one which effects page transfers only on the occurrence of a page fault. Such a policy is termed demand pre-paging [TRIV74]. The optimal demand pre-paging policy, DPMIN, would, on the occurrence of a fault, update the contents of the buffer so as to contain the n pages with the most imminent references. This policy, like MIN, is unrealizable, but serves as a standard of reference.

The analysis of this algorithm is simplified by the use of the Most Imminent Reference Stack Model (MIRSM). In the MIRSM, the pages are ordered by imminence of reference. The page at the top of the MIR stack is the one which will be referenced next. After a page is referenced, it is inserted in position i of the MIR stack with probability p(i) and all the pages which were in positions 2 through i are moved up one position. p(i) is the probability that the page will next he referenced after i distinct pages (those above it in the MIR stack) have been referenced. So, the p(i) of the MIRSM and the p(i) of the LRUSM are identical. Fig.8 shows the Markov chain corresponding to the path of a page through the MIR stack.

Returning to the DPMIN policy, let us assume that a page fault has just occurred. Consequently, the top n pages in the MIR stack are all

in the buffer. The next fault occurs on the first occasion that a page in positions n+l through N is referenced. Such a page must necessarily be in position n+l since this is the page which will be referenced before any of the other pages which are not in the buffer. We wish to determine the mean time for a page to travel from position n+l to position 1 in the MIR stack.

From Fig.8 we see that the mean time spent in position i before moving on to position i-l is given by l/m(i-1) for $1<i\leq N$. Therefore, the mean time between faults is

$$\sum_{i=1}^{n} 1/m(i)$$

The mean fault rate is then given by

$$1/\sum_{i=1}^{n} 1/m(i)$$

It is possible to obtain an upper bound on the page traffic too. With the DPMIN policy, a page is replaced if, on the occurrence of a fault, it is at a position in the MIR stack lower than position n. This event takes place if, after a reference, it is inserted at a position lower than n and it does not rise into the top n positions before the next fault. If we neglect the second condition, we over estimate the probability of a page being replaced and, accordingly, overestimate the traffic.

Once again, let us assume that a fault has just occurred and that the top n pages are present in the buffer. Let us also assume that the page in position n+1 is tagged. While the tagged page stays in position n+1, all referenced pages are, necessarily, being inserted into positions 1 through n. The tagged page rises one position only when the referenced page is inserted in the position currently occupied by the tagged page or in a position lower than that. Conversely, a page can be inserted into positions n+1 and lower (and be replaced) only when the tagged page rises one position.

If the tagged page rises from position i to i-1, $(1<i\leq n+1)$, then the referenced page must have been inserted in one of the positions i through N. Conditional on this, the probability that it was inserted below n is

$$[p(n+1)+p(n+2)+\ldots+p(N)]/[p(i)+p(i+1)+\ldots+p(N)] = m(n)/m(i-1).$$

So, the average number of pages replaced when the tagged page rises from position i to i-1 is m(n)/m(i-1). The average number of pages replaced per fault using DPMIN is bounded above by

$$\sum_{i=1}^{n} m(n)/m(i) = m(n) \sum_{i=1}^{n} 1/m(i)$$

The traffic is given by the mean number of pages replaced per fault multiplied by the fault rate, and is bounded above by

$$m(n)\left[\sum_{i=1}^{n} 1/m(i)\right]\left[1/\sum_{i=1}^{n} 1/m(i)\right] = m(n)$$

which is the traffic generated by the LRU policy.

We see then that DPMIN can reduce the fault rate considerably compared to LRU without increasing the traffic.

## 6.3 Sharing of Memory using the Global LRU policy

Lastly, we consider a buffer shared by a number of concurrently active processes. Such a situation might arise in a multiprocessor system where all the processors share a common cache or main memory, or in a multiple instruction stream processor with shared resources [FLYN68]. In any event it is assumed that the next request will be from process i with probability $r_i$. A multiprogramming system cannot be included in this framework since only one process is active at any time. The memory management policy is assumed to be Global LRU. We shall use the Stack Working Set to determine the manner in which the capacity of the shared buffer is allocated to the individual processes, i.e., if d is the capacity of the shared buffer and we have q concurrent processes, we shall calculate $\{d_i\}$, where di is the mean number of pages of process i present in the buffer and $(d_1+d_2+\ldots+d_q) = d$.

At time t, the SWS for process i has a size of $D_i(t)$. The total space occupied by all the SWS's at time t is given by

$$D(t) = \sum_{i=1}^{q} D_i(t)$$

Let $t_0$ be the value of t such that $D(t_0)=d$. Then $d_i=D_i(t_0)$. $t_0$ is easily obtained either graphically or by inverting D(t).

## 7 Conclusion

We have derived a number of properties of the LRUSM. Most of these are exact results. It is worth our while to list the approximations that we have used and to classify them. In developing and using the Stack Working Set, we have made the conceptual approximation that it and Denning's Working Set can be treated identically. We also used a mathematical approximation in allowing ourselves to treat discrete functions as continuous ones by interpolation. We used another conceptual approximation when we relaced the random variables for the replacement and residency times by constants equal to the means. This approximation turned out to be poor for large page sizes.

REFERENCES

BELA66 L.A.Belady, "A Study Of Replacement Algorithms For A Virtual-Storage Computer", IBM Sys. Jour., 5, 2, 78-101, 1966.

COFF73 E.G.Coffman and P.J.Denning, "Operating Systems Theory", Prentice-Hall, 275-278, 1973

DENN68 P.J.Denning, "The Working Set Model for Program Behavior", CACM 11,5, 323-333, May 1968.

DENN72 P.J.Denning and S.C.Schwartz, "Properties of the Working Set Model", CACM 15,3, 191-198, Mar.1972.

FELL68 W.Feller, "An Introduction to Probability Theory and Its Applications", Vol.I, 3rd Edition, John Wiley, 264-301, 1968.

FLYN68 M.J.Flynn, A.Podvin and K.Shimizu, 'A Multiple Instruction Stream Processor with Shared Resources", Proc. Conf. on Parallel Processing, Monterey, California, 1968; Pub. Spartan Press 1970.

FRAN74 P.A.Franaszek and T.J.Wagner, "Some Distribution- Free Aspects Of Paging Algorithm Performance"JACM 21, 1, 31-39, Jan 1974.

KAPL73 K.R.Kaplan and R.O.Winder, "Cache-Based Computer Systems", Computer, 6, 3, 30-36, Mar. 1973.

KING71 W.F.King, 'Analysis of Demand Paging Algorithms", Proc. IFIP Congress 1971, Ljubljana. Amsterdam: North-Holland, TA-3 155-162, 1971.

MATT70 R.L.Mattson, J.Gecsei, D.R.Slutz and I.L.Traiger, "Evaluation Techniques for Storage Hierarchies', IBM Sys. Jour., 9, 2, 1970

RAF176 A.Rafii, "Empirical and Analytical Studies of Program Reference Behavior", Stanford Linear Accelerator Tech. Rep. No. 197, July 1976.

RAo75 G.S.Rao, 'Performance Evaluation of Cache Memories", Stanford Univ., Dept. Elec. Engg., Ph.D. Thesis, 1975.

RAU75 B.R.Rau, "The Stack Working Set: A Characterization of Spatial Locality", Tech. Rep. No. 95, Digital Systems Lab., Stanford Univ., July 1975.

SHEM66 J.E.Shemer and G.A.Shippey, "Statistical Analysis of Paged and Segmented Computer Systems", IEEE-TEC, EC-15,6, 855-863, Dec.1966.

TRIV74 K.S.Trivedi, "Prepaging and Applications to Structured Array Problems", Tech.Rep.No. UIUCDCS-R-74-662, Dept. of Comp. Sci., Univ. of Illinois at Urbana-Champaign, July 1974.

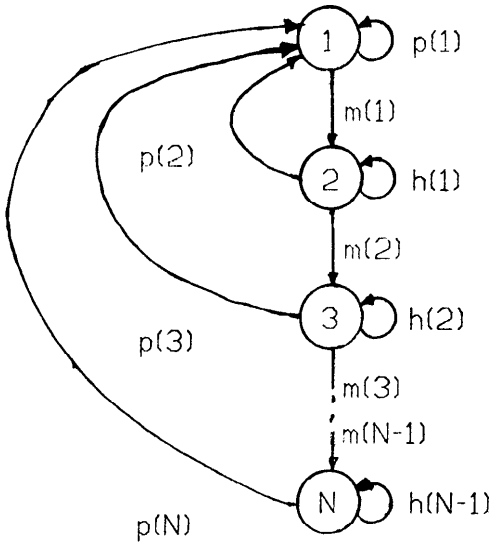
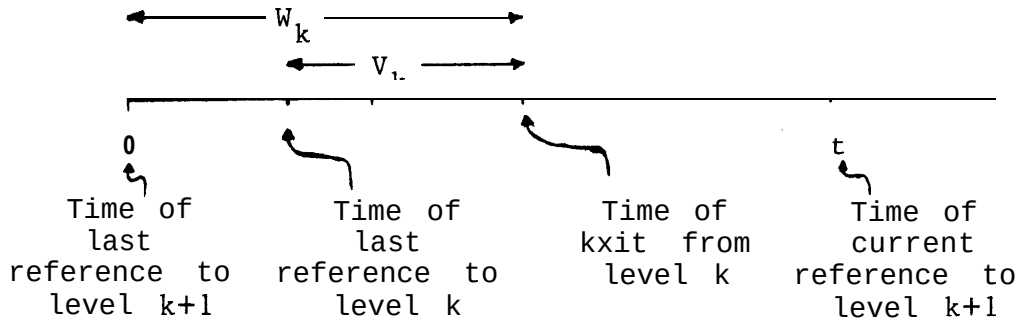



FIG. 1

TIME BETWEEN SUCCESSIVE REFERENCES AT LEVEL K+1




FIG, 2

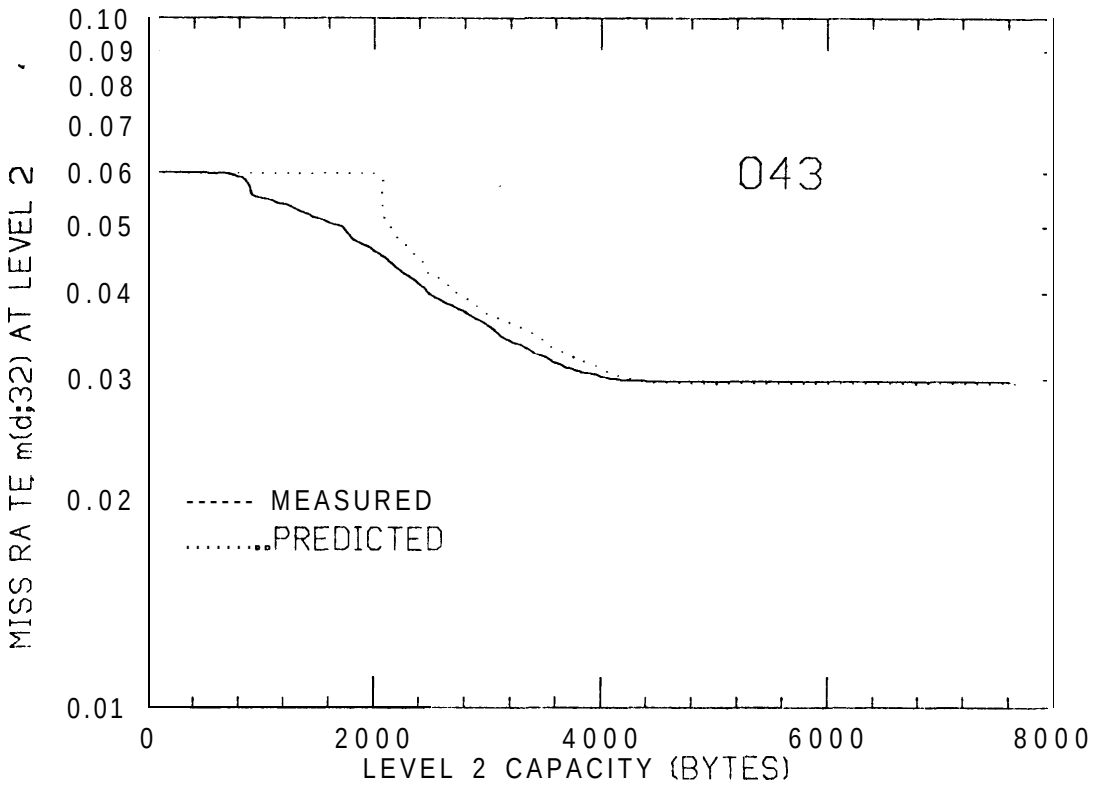

PREDICTED AND MEASURED FAULT STREAMS
0.10
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
MISS RA TE m(d;32) AT LEVEL 2
043
------ MEASURED
.....PREDICTED
0
2000
4000
6000
8000
LEVEL 2 CAPACITY (BYTES)

FIG. 3

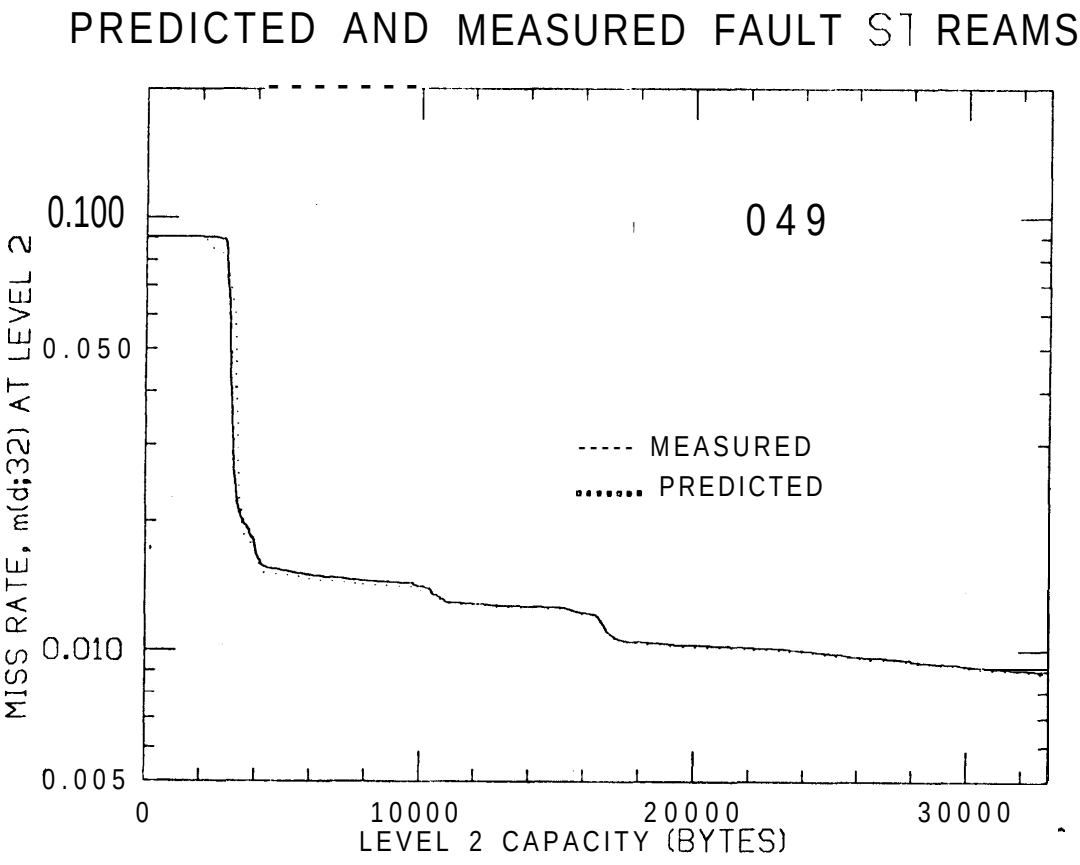

PREDICTED AND MEASURED FAULT ST REAMS
0.100
0.050
0.010
0.005
MISS RATE, m(d;32) AT LEVEL 2
049
----- MEASURED
..... PREDICTED
0
10000
20000
30000
LEVEL 2 CAPACITY (BYTES)

FIG. 4

PREDICTED AND MEASURED FAULT STREAMS

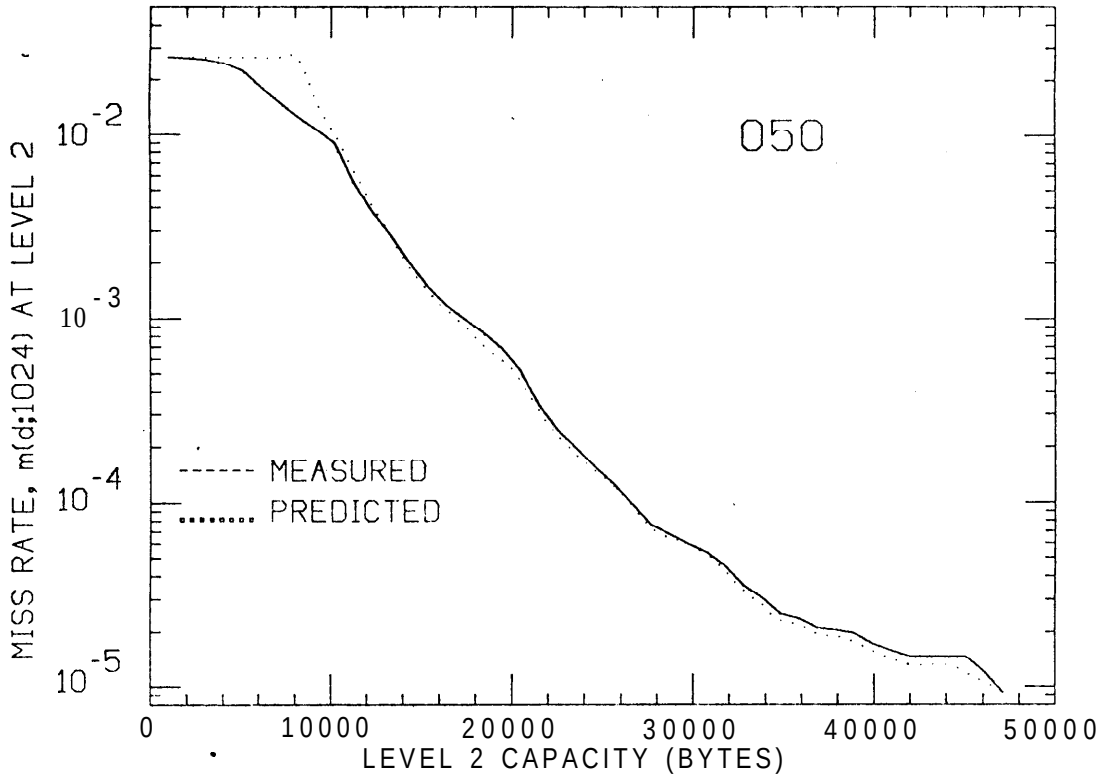
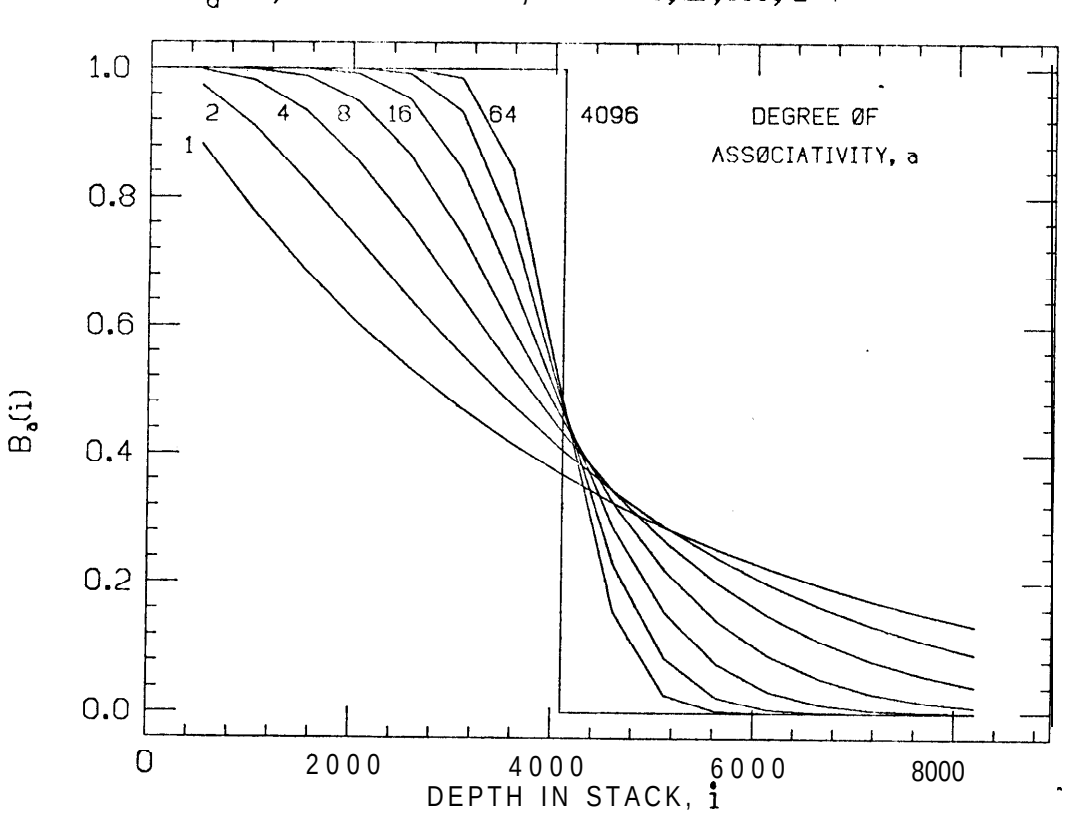



FIG. 5




FIG, 6

LRUSM STATISTICS WHICH GIVE ANØMALØUS RESULTS

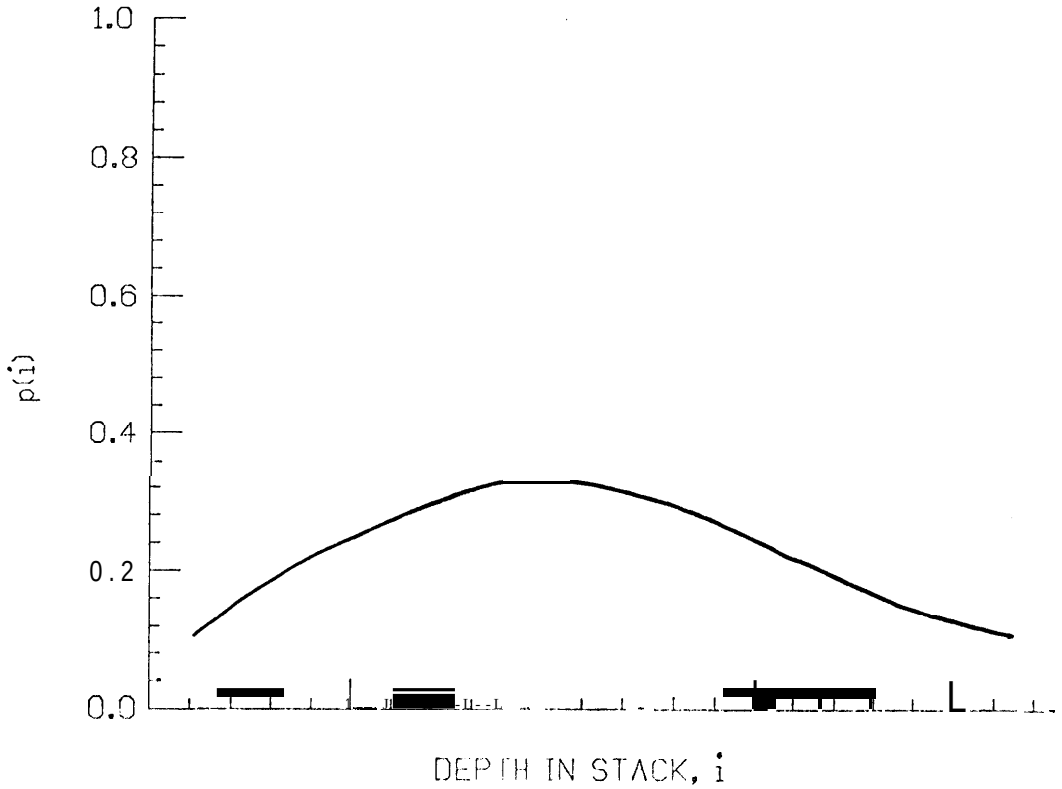



FIG. 7

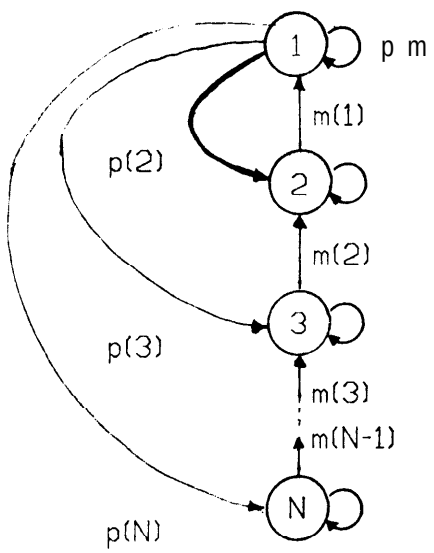



FIG. 8