Master of Science Degree Project Report
University of California, Berkeley,
Berkeley, California 94720

ANALYSIS OF BUFFERING IN MEMORY INTERLEAVING

Committee: Professor C.V. Ramamoorthy
Professor L.A. Zadeh

Benjamin W. Wah

June, 1976.

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitude to Professor C.V. Ramamoorthy and Professor L.A. Zadeh who have contributed in every way to the development of this project in Berkeley. Through their constant advice, support, guidance and encouragement, especially at time when the inevitable evil of disillusionment arose, this project has finally come into shape. Without their help, none of these would have been possible.

He also wishes to express his gratitude to both the National Science Foundation and the U.S. Army Research Office at Durham for their support through grants NSF GJ-35839 and ARODA-ARO-D-31-124-73-G157 under which the reported work has been accomplished.

Finally, the author wishes to acknowledge the guidance and suggestion of Mr. C.R. Vick and Dr. C.G. Davis of the U.S. Army BMD-ATC, Huntsville, Alabama, and Mr. Phil Moore of the System Development Corporation, Huntsville. The program traces supplied by them made possible the simulation in this project.

INTRODUCTION

Memory interleaving is a form of pipelining to increase the speed of memory access. It may consists of say m memory units each taking m processor cycles to deliver a word. To avoid undue extra cost, no actual memory replication is used; only the addresses are interleaved so that all addresses equalling (j modulo m) refers to memory box j. This way the input stream of addresses now behaves like an admixture of m different streams; nevertheless consecutive addresses refer to different boxes, and the one-output-per-cycle maximum rate can be honored for well chosen address request sequences.

A multiple module system has two promising characteristics for high speed operation. First, for a given capacity, a smaller module can usually be designed with faster cycle time because propagation and other delays can be smaller. Second and most important, a multiple module system permits several modules to be accessed at one time.

Because of program locality, successive accesses are often made to consecutively numbered addresses. For this reason, in multiple module systems, successive addresses are assigned to successive modules.

The normalized performance of an interleaved memory system is P_{int} = (throughput of the interleaved system) / (throughput of the single box memory). The limiting cases of module concurrency is 1 when all successive calls are to the same module and a maximum concurrency of m when certain fortuitous sequences occur. Hallerman has shown that for random requests, the average throughput in a first-come, first-served system is

$$N_{avg} = \sum_{k=1}^{m} \frac{k^2 (m-1)!}{h^k (m-k)!} \quad , \qquad [1]$$

A fairly good approximation is $N_{avg} = m^{0.56}$, $1 \leq m \leq 45$. A better approximation is by using Knuth's Q function. It is shown that

$$N_{avg} = \left(\frac{\pi m}{2}\right)^{\frac{1}{2}} - \frac{1}{3} + \frac{1}{12}\left(\frac{\pi}{2m}\right)^{\frac{1}{2}} - \frac{4}{135} m^{-1} + O(m^{-\frac{3}{2}}) \qquad [2]\ [3]$$

A different model is investigated in [4] which allows queuing on busy models. The expected value for the bandwidth is

$$E(B) = \sum_{k=1}^{t} \frac{k.k!S(p,k)\binom{m}{k}}{m^p} \quad \text{where } t = \begin{cases} p & p \leq m \\ m & p > m \end{cases}$$

and p is the number of processors making requests. The bandwidth obtained are much higher than those expected from Hellerman's model. That model assumes that there are at least as many requests in the request sequence as there are memory modules. Further, the assumption of random requests in both models is not realistic. Although it is difficult to say anything general about program behaviour, it can be said that "real programs are not random in their addressing patterns" [5]. If such is the case, the

expected value for the bandwidth would be actually higher than these two models predict.

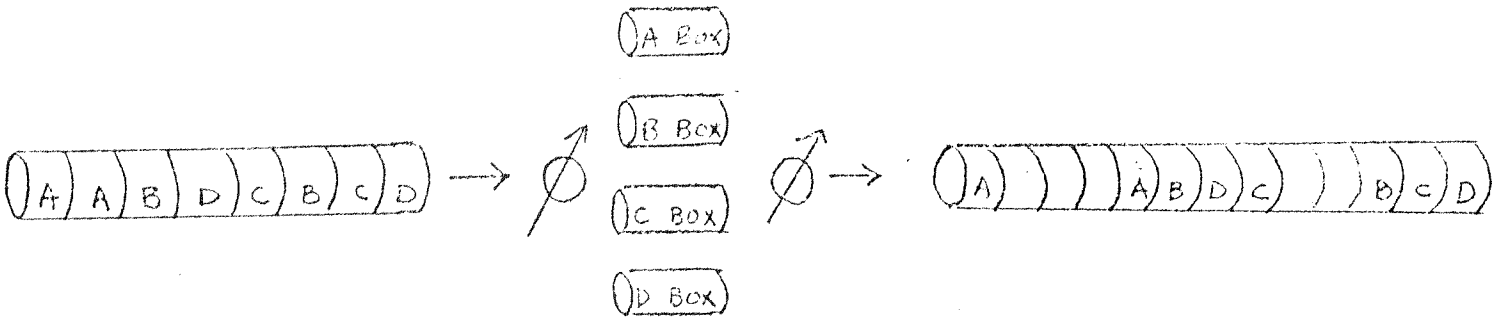
The average predicted by the previous models can be improved by buffering the request sequence and honoring only those requests which refer to boxes currently available [6]. Fig. 1 shows the dramatic improvement with only two buffer registers. The system, with the queue selection mechanism, optimizes its own throughput.

Typically, the ordering of the delivery sequence does not mirror the input sequence and detailed prediction of system behaviour becomes difficult. The gross behaviour, on the other hand, now tends to exhibit a good local statistical distribution.

The sequence can be resorted using a set of buffer registers; there will, however, additional delays. Whether resorted or not, there must be temporary identifiers associated with the delivered words to compensate for the out-of-sequence delivery.

This project will try to solve analytically the improvement in performance on different amount of buffering. Simulations will be done on the address traces of a program compilation and execution.

a) Memory Interleaving based on first come, first served discipline.



b) Buffer memory associated with the switching. The delivery is out of sequence

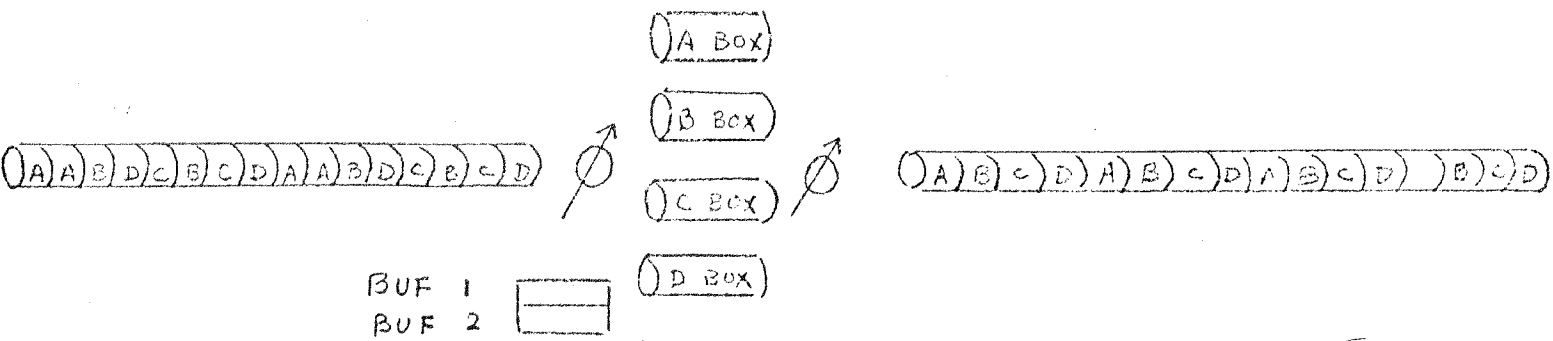


FIG. 1 MEMORY INTERLEAVING

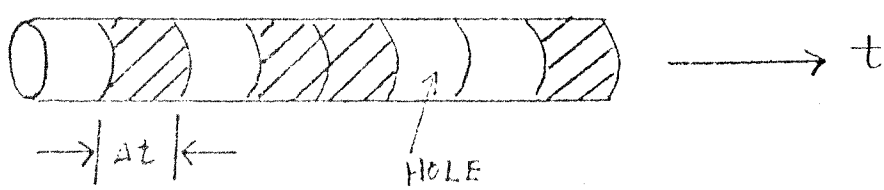


FIG 2 FULLNESS IN OUTPUT STREAM

DEFINITION OF IMPROVEMENT IN PERFORMANCE

Assume that there is a memory system with m ways of interleaving and r buffers lookahead. The efficiency of this buffering scheme can be put in terms of the "fullness" of the output stream. The output stream can be considered as a function of time and is divided into time intervals Δt, where Δt is the time between "sweeping" of adjacent memory modules. If data is being output during a time interval, then that time interval is "full", otherwise, a "hole" exists (Fig. 2).

The "fullness" in the output stream is defined by F(r,m)

$$F(r,m) = 1 - \frac{\text{number of holes in output stream}}{\text{total number of } t \text{ in output stream}}$$

We define that S_1 to be the state of the memory system which can make a sucessful reference and S_2 to be the state which will give a hole in the output stream. S_1 and S_2 are gross states of the machine. Internal states of S_1 and S_2 consist of the state of the buffer, the current reference in the request sequence and the module currently referencing. The state of a buffer is the maximum number of memory modules it has to traverse before a sucessful reference can be made. The state of a buffer will therefore give the probability that a sucessful reference can be made in the current sweep. We also define $\#(S_1)$ and $\#(S_2)$ to be the number of occurences in states S_1 and S_2 respectively. Therefore

$$F(r,m) = \frac{\#(S_1)}{\#(S_1) + \#(S_2)}$$

The improvement in performance is defined as the ratio of

fullness in output stream with buffering and fullness without buffering, that is

$$IMP(r,m) = F(r,m)/F(0,m)$$

The maximum improvement is clearly when the amount of buffering is infinite.

$$IMP_{max}(m) = F(\infty,m)/F(0,m) = 1.0/F(0,m)$$

The relative improvement is defined as

$$RELIMP = (IMP(r,m) - IMP(0,m))/(IMP_{max}(m) - IMP(0,m))$$

since

$$IMP(0,m) = F(0,m)/F(0,m) = 1.0$$

Therefore

$$RELIMP = (IMP(r,m) - 1)/(IMP_{max}(m) - 1)$$

TOOLS IN ANALYSIS

Internal states in S_1 and S_2 can be defined as a transition from one to another. A state diagram can therefore be drawn. The size of the state diagram depends on both r and m. r will define the number of state variable. m will define the number of extent of each state variable. The number of states is therefore proportional to m^r. However, we do not try to distinguish between states like (a,b) and (b,a). The number of states is therefore slightly less.

In state S_1, the number of buffers can define the number of state variables. The current state of the memory reference in the request sequence and the state of the memory sweep can be used as an input variable. In state S_2, the number of state variable equals the number of buffers plus one. The way that the system can go into state S_2 is because a current reference cannot be satisfied. This reference will therefore add an extra state variable into S_2. The state of the memory sweep is similiarly used as an input variable.

The mapping of each state into an unique integer is similiar to the mapping of an r-dimensional tetrahedral array into sequential storage. A solution for the latter mapping is given in [7].

Mapping in S_1 is

$$M(s_1,s_2,\ldots,s_r) = M(0,0,\ldots,0) + \sum_{1\leq k\leq r} \binom{s_k + r - k}{1 + r - k}$$

and $\quad 0 \leq s_r \leq s_{r-1} \leq \ \ldots \ \leq s_1 \leq m-1$

Mapping in S_2 is

$$M'(s_1,s_2,\ldots,s_r,s_{r+1}) = M'(1,1,..1) + \sum_{1\leq K\leq r+1} \binom{s_k+r-k}{2+r-k}$$

We can assume for convenience that $M(0,0,\ldots,0) = M'(1,1,\ldots,1) = 1$. There is also an extra state, the fail state in S_1 and the success state in S_2.

The total number of states is therefore:

for S_1, number of states $= 2 + \sum_{1 \leq k \leq r} \binom{m - 1 + r - k}{1 + r - k}$

for S_2, number of states $= 2 + \sum_{1 \leq k \leq r+1} \binom{m - 1 + r - k}{2 + r - k}$

Fig. 3 shows a table of different number of states vs. r and m.

We see that the number of states in S_2 increases less rapidly than the number of states in S_1. However, the number of states in S_2 is larger than the number of states in S_1 initially.

The transition from one state to another can be represented in a two dimensional transition matrix. But since the representation of a 2 dimensional matrix in a digital computer is limited to the order of 100 x 100, it is difficult to solve analytically for large number of register lookahead.

By the use of transition matrix where element $T(i,j)$ is the probability of transition from state i to state j, and setting the fail state (in the case of S_1) or the success state (in the case of S_2) as the absorbing state, we can use Markov chain theory to calculate the number and variance of steps in which the process in a transient state for such an absorbing chain [8].

The transition matrix is of the following form:

...

...

MDM	RDG	# OF STATES IN S_1	# OF STATES IN S_2
2	1	3	2
	2	4	2
	3	5	2
	4	6	2
	5	7	2
	6	8	2
	7	9	2
4	1	5	7
	2	11	11
	3	21	16
	4	36	22
	5	57	29
	6	85	37
	7	121	46
8	1	9	29
	2	37	85
	3	121	211
	4	331	463
	5	793	925
	6	1717	1717
	7	3433	3004
16	1	17	121
	2	137	681
	3	817	3061
	4	3877	11629
	5	15505	38761

Fig. 3 Table of # of States VS r and n

success/fail	s_1	s_2	...	s_k
success/fail	1	0	0 ...	0
s_1	x			
s_2	x			
.	.		Q	
.	.			
.	.			
s_k	x			

where Q is a sequence matrix of size k * k. For an absorbing Markov chain, the fundamental matrix is defined as $N = (I - Q)^{-1}$ where I is the identity matrix of size k * k.

We define n_j to be the function giving the total number of times that the process is in s_j (this is defined only for transient state s_j) and $M_i(n_j)$ and $Var_i(n_j)$ is the mean and variance of the function n_j when the chain is started in s_i.

It is shown in [8] that

1) $\{M_i(n_j)\} = N$ and

2) $\{Var_i(n_j)\} = N(2N_{dg} - I) - N^2$

where N_{dg} is an K x K matrix consisting of only the diagonal elements of matrix N.

We further define t as the function giving the number of steps (including the original position) in which the process is in a transient state.

It is also shown in [8] that

1) $\{M_i(t)\} = N\xi = \tau$

2) $\{Var_i(t)\} = (2N - I)\tau - \tau^2$

where ξ is a column matrix of size K, with all elements equal 1.

If π is the initial probability vector for an absorbing chain, and π' consists of the last K components of π, i.e. π' gives the initial probabilities for the transient states, then,

$$\{M_{\pi}(n_j)\} = \pi' N$$

$$\{\mathrm{Var}_{\pi}(n_j)\} = \pi' N(2N_{dg} - I) - (\pi' N)^2$$

$$\{M_{\pi}(t)\} = \pi' N \xi = \pi' \tau$$

$$\{\mathrm{Var}_{\pi}(t)\} = \pi'(2N - I)\tau - (\pi'\tau)^2$$

Utilizing these properties in absorbing Markov chains, we can then represent the transition matrices in S_1 and S_2 and calculate the number of steps before it fails or succeeds. The means obtained will be $\#(S_1)$ and $\#(S_2)$ respectively.

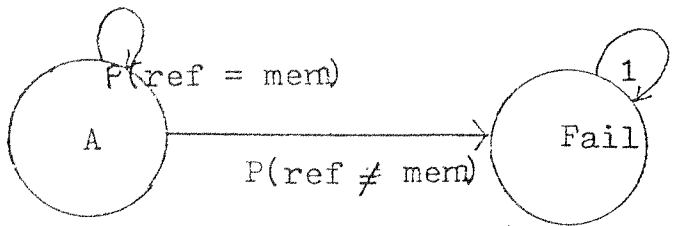
GENERATION OF TRANSITION MATRIX FOR S_1

In the generation of states for S_1, some of the states have been ignored. These states should originally be generated in the following ways: let there be m ways of interleaving, for the state (a, b, c), if two of the buffer contents equal the currently referenced module, say b and c, then b will be referenced and c will take another m-1 turns before it can be referenced again. Therefore the next state should be (a-1, 0) instead of (a-1, 0, m-1). The state (a-1, 0) will undergo m-1 transitions in states with two state variables before it goes back to states with three variables. However, this will add an enormous amount of states to the system and will complicate the system unnecessarily. If we make an assumption that we do not keep track of whether this can happen, then the transition to state (a-1, 0, m-1) instead of (a-1, 0) can be justified. This will reduce the number of states used.

Exmaples will first be shown for the cases of a) r = 0, b) r = 1, m = 4, and c) r = 2, m = 4. I will attempt to reduce the number of states by lumping. A general algorithm for generating the transition matrix P_1 for S_1 will also be discussed.

Example (a) r = 0

The state diagram can be represented as



where ref = current reference in the request sequence

mem = currently referenced memory module.

The transition matrix

$$P_1 = \begin{array}{c} \\ \text{Fail} \\ A \end{array} \begin{array}{c} \begin{array}{cc} \text{Fail} & A \end{array} \\ \begin{bmatrix} 1 & 0 \\ 1 - \frac{1}{m} & \frac{1}{m} \end{bmatrix} \end{array}$$

$$Q = [1/m]$$

$$N = (I - Q)^{-1} = (1 - 1/m)^{-1} = [m/(m - 1)]$$

$$\bar{\pi}' = [1]$$

$$\{M_{\pi}(t)\} = \bar{\pi}'(N\xi) = m/(m - 1)$$

$$\{Var_{\pi}(t)\} = \bar{\pi}'(2N - I)[- (\bar{\pi}'\xi)^2 = m^2/(m - 1)^2$$

eg. when m = 4

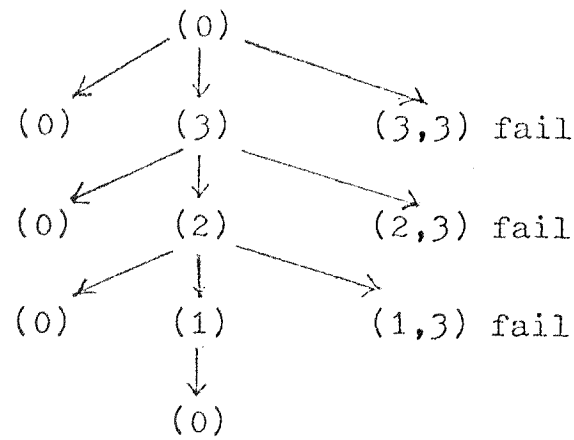
$$\{M_{\pi}(t)\} = 4/3 = 1\tfrac{1}{3}$$

$$\{Var_{\pi}(t)\} = (4/3)^2 = 1\tfrac{7}{9}$$

Example (b) r=1, m=4

The buffer can be in one of the four states (0), (1), (2), or (3). There is an extra state, the fail state.

The transition in states can be represented in the form of a tree.



Recall that the state of the buffer is the maximum number of memory modules that the reference has to compare before a match can be found. The failing state is of some interest; it is a 2-tuple instead of a 1-tuple. The 2nd state variable added is always m-1 (m=4 in this example). The process fails because there is an extra reference in the request sequence and the contents of the buffer cannot be matched with the currently referenced module. States (1,3), (2,3) and (3,3) therefore are the only states that the process can start in S_2.

initial state	final state	explanation	probability
(0)	(0)	ref_1 = mem	1/4
(0)	(3)	$ref_1 \neq$ mem, ref_2 = mem	3/4 . 1/4 = 3/1
(0)	(3,3) fail	$ref_1 \neq$ mem, $ref_2 \neq$ mem	3/4 . 3/4 = 9/1
(3)	(0)	buf = mem	1/3
(3)	(2)	buf $\neq$ mem, ref_1 = mem	2/3 . 1/4 = 1/6
(3)	(2,3) fail	buf $\neq$ mem, $ref_1 \neq$ mem	2/3 . 3/4 = 1/2
(2)	(0)	buf = mem	1/2
(2)	(1)	buf $\neq$ mem, ref_1 = mem	1/2 . 1/4 = 1/8
(2)	(1,3) fail	buf $\neq$ mem, $ref_1 \neq$ mem	1/2 . . 3/4 = 3/8
(1)	(0)	buf = mem	1

The transition matrix can be put together as:

$$P_1 = \begin{array}{c} \\ (0) \\ (1) \\ (2) \\ (3) \\ \text{Fail} \end{array} \begin{array}{c} \begin{array}{ccccc} (0) & (1) & (2) & (3) & \text{Fail} \end{array} \\ \begin{bmatrix} 1/4 & 0 & 0 & 3/16 & 9/16 \\ 1 & 0 & 0 & 0 & 0 \\ 1/2 & 1/8 & 0 & 0 & 3/8 \\ 1/3 & 0 & 1/6 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ (3) \\ (2) \\ (1) \\ (0) \end{array} \begin{array}{c} \begin{array}{cccc} (3) & (2) & (1) & (0) \end{array} \\ \begin{bmatrix} 0 & 1/6 & 0 & 1/3 \\ 0 & 0 & 1/8 & 1/2 \\ 0 & 0 & 0 & 1 \\ 3/16 & 0 & 0 & 1/4 \end{bmatrix} \end{array}$$

$$N = (I - Q)^{-1} = \frac{256}{171}\begin{bmatrix} 3/4 & 1/8 & 1/64 & 7/16 \\ 15/128 & 11/16 & 11/128 & 5/8 \\ 3/16 & 1/32 & 43/64 & 1 \\ 3/16 & 1/32 & 1/256 & 1 \end{bmatrix}$$

$$= N3 = \frac{256}{171}\begin{bmatrix} 85/64 \\ 97/64 \\ 121/64 \\ 313/64 \end{bmatrix}$$

We can assume that $\pi'_{S_1} = (0\ 0\ 0\ 1)$ which means that it starts initially in state (0). (The calculation of initial probability vector π' is investigated later).

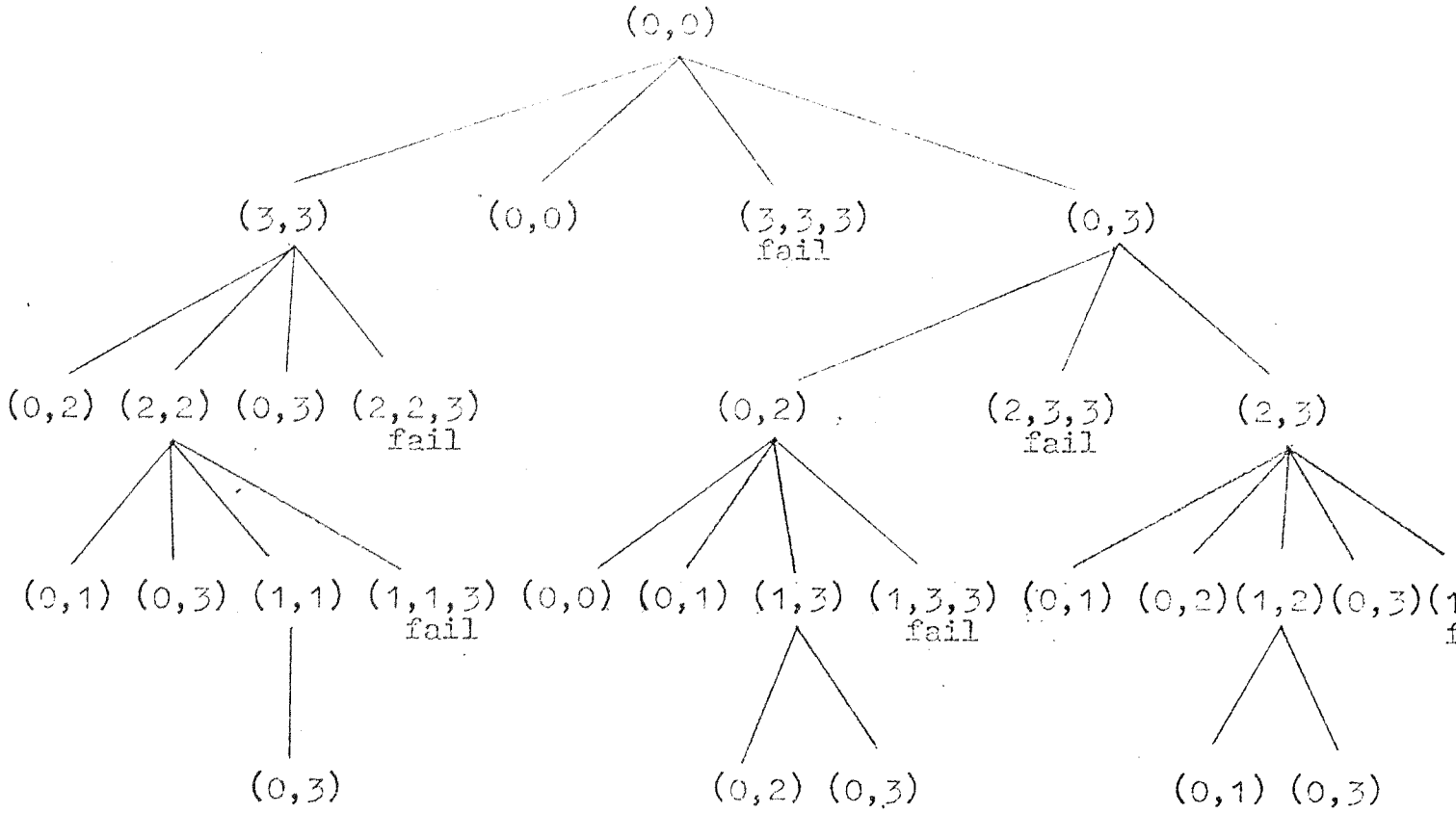
$$\{M_{\pi', s_1}(t)\} = 256/171 * 313/256 = 313/171 = 1.83$$

$$\{Var_{\pi', s_1}(t)\} = 5.01 - (1.83)^2 = 1.67$$

Example (c) r =2, m = 4

The buffers can be in one of the 10 states: (0,0), (0,1), (0,2), (0,3), (1,1), (1,2), (1,3), (2,2), (2,3), (3,3). There is an extra state, the fail state.

The transition in states can be represented as:



The interpretation of the states is the same as before.

initial state	final state	explanation	probability
(0,0)	(0,0)	ref_1 = mem	1/4
	(0,3)	$ref_1 \neq$ mem, ref_2 = mem	3/4 . 1/4 = 3/16
	(3,3)	$ref_1 \neq$ mem, $ref_2 \neq$ mem, ref_3 = mem	3/4 . 3/4 . 1/4 = 9/64
	(3,3,3) fail	$ref_1 \neq$ mem, $ref_2 \neq$ mem, $ref_3 \neq$ mem	3/4 . 3/4 . 3/4 = 27/64
(0,1)	(0,0)	buf = mem	1
(0,i)	(0,0)	buf = mem	1/i
i > 1	(0,i-1)	buf $\neq$ mem, ref_1 = mem	(1 - 1/i)·1/4
	(i-1, 3)	buf $\neq$ mem, $ref_1 \neq$ mem, ref_2 = mem	(1 - 1/i)(3/4) (1/4)
	(i-1, 3,3) fail	buf $\neq$ mem, $ref_1 \neq$ mem, $ref_2 \neq$ mem	(1-1/i) (3/4) (3/4)
(1,1)	(0,3)	buf = mem	1
(1,i)	(0,3)	buf_1 = mem, buf_2 = mem	1/i
i > 1	(0,i-1)	buf_1 = mem, $buf_2 \neq$ mem	1 - 1/i
(i, j)	(0,i-1)	buf_1 = mem, $buf_2 \neq$ mem	(1/i) (1- 1/j)
i,j 1	(0,j-1)	$buf_1 \neq$ mem, buf_2 = mem	(1/j) (1 - 1/i)
	(0,3)	buf_1 = mem, buf_2 = mem	(1/i) (1/j)
	(i-1,j-1)	buf $\neq$ mem, ref_1 = mem	(1-1/i) (1-1/j) (1/4)
	(i-1,j-1,3) fail	buf $\neq$ mem, $ref_i \neq$ mem	(1-1/i)(1-1/j)(3/4)

The transition matrix is

	(0,0)	(0,1)	(0,2)	(0,3)	(1,1)	(1,2)	(1,3)	(2,2)	(2,3)	(3,3)	fail
(0,0)	1/4	0	0	3/16	0	0	0	0	0	9/64	27/64
(0,1)	1	0	0	0	0	0	0	0	0	0	0
(0,2)	1/2	1/8	0	0	0	0	3/32	0	0	0	9/32
(0,3)	1/3	0	1/6	0	0	0	0	0	1/8	0	3/8
(1,1)	0	0	0	1	0	0	0	0	0	0	0
(1,2)	0	1/2	0	1/2	0	0	0	0	0	0	0
(1,3)	0	0	2/3	1/3	0	0	0	0	0	0	0
(2,2)	0	1/2	0	1/4	1/16	0	0	0	0	0	3/16
(2,3)	0	1/6	1/3	1/6	0	1/12	0	0	0	0	1/4
(3,3)	0	0	4/9	1/9	0	0	0	1/9	0	0	1/3
fail	0	0	0	0	0	0	0	0	0	0	1

Reduction of the number of states by State Lumping

The number of states involved is very large and increases exponentially with r and m. If some of the states can be lumped together, the size of the matrix can be reduced drastically. Therefore it is advantageous to investigate the possibility of lumping states.

A set of states can be partitioned into N sets (A1,A2 . . An) if for any 2 sets A_i, A_j. P_{ik} A_j are the same for every state S_k in Ai ($P_{ij} = \sum_{s \in A_j} P_{kn}$) [9]. Since we are dealing with an absorbing chain.

we therefore want to lump states of the same kind, that is, absorbing states with absorbing ones, and non-absorbing states with non-absorbing ones.

A matrix used for the calculation of S_1 for the case n=4 r=3 is shown in Fig. 4. The last state is the absorbing state. The non-zero entries in that column are all different, therefore none of them can be lumped together. For all of the zero entries, an exhaustive search can be made to determine whether any 2 states can be combined. It is seen that for this case, none can be com combined.

In general because of the sparseness of the matrix, none (or very few, if any) of the states can be lumped together.

An algorithm to generate the transition matrix for S_1

This algorithm will generate the transition probabilities for the different types of states in S_1. The states will be grouped together in such a way so that the transition probability can be generated by a single equation for each type.

a) States of the form (0,...,0,1,...,1)
(the number of 1's in the state is h)
The next state will be (0,...,0,n-1,n-1,...) where the number of m-1's is h-1 with a transition probability of 1.

b) States of the form $(1,\ldots,1,i_1,i_2,\ldots,i_k)$.
where the number of 1's is h, and $1<i_1\leq i_2\leq i_3 \ldots\leq n-1$, $i_1,i_2,\ldots,i_k$ will be enumerated for zero to k combinations of i_j's where $1\leq j\leq k$. The resulting set of states are there

TRANSITION MATRIX FOR S1, (R=3, M =4)

.250	0.	0.	0.	0.	0.	0.	0.	0.	0.	.188	0.	0.	0.	0.	0.	.141	0.	0.	.105	.316
1.000	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000	0.	0.	0.	0.
.500	.125	0.	0.	0.	0.	0.	0.	0.	0.	0.	.094	0.	0.	0.	0.	0.	.070	0.	0.	.211
0.	.500	0.	0.	0.	0.	0.	0.	0.	0.	.500	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.500	0.	0.	0.	0.	.500	0.	0.	0.	0.
0.	.500	.063	0.	0.	0.	0.	0.	0.	0.	.250	0.	.047	0.	0.	0.	0.	0.	0.	0.	.141
0.	0.	.250	0.	0.	0.	0.	0.	0.	0.	0.	.500	0.	0.	0.	0.	.250	0.	0.	0.	0.
0.	0.	.375	.031	0.	0.	0.	0.	0.	0.	0.	.375	0.	0.	0.	0.	.125	0.	0.	0.	.094
.333	0.	0.	0.	.167	0.	0.	0.	0.	0.	0.	0.	0.	.125	0.	0.	0.	0.	.094	0.	.281
0.	0.	0.	0.	.667	0.	0.	0.	0.	0.	.333	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.667	0.	0.	.333	0.	0.	0.	0.
0.	.167	0.	0.	.333	.083	0.	0.	0.	0.	.167	0.	0.	0.	.062	0.	0.	0.	0.	0.	.187
0.	0.	0.	0.	0.	.333	0.	0.	0.	0.	0.	.167	0.	.333	0.	0.	.167	0.	0.	0.	0.
0.	0.	.083	0.	0.	.333	.042	0.	0.	0.	0.	.167	0.	.167	0.	0.	.083	0.	0.	0.	.125
0.	0.	0.	0.	.444	0.	0.	.111	0.	0.	.111	0.	0.	0.	0.	.083	0.	0.	0.	0.	.250
0.	0.	0.	0.	0.	0.	0.	.444	0.	0.	0.	0.	0.	.444	0.	0.	.111	0.	0.	0.	0.
0.	0.	0.	0.	0.	.222	0.	.222	.056	0.	0.	.056	0.	.222	0.	0.	.056	0.	0.	0.	.167
0.	0.	0.	0.	0.	0.	0.	.444	0.	.074	0.	0.	0.	.222	0.	0.	.037	0.	0.	0.	.222
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000

TRANSITION MATRIX FOR S2 (R=3, M=4)

0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
.063	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.938
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	.083	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.917
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	.111	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.889
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000
0.	0.	0.	.148	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.852
0.	0.	0.	0.	.198	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	.802
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.000

buffered references which equal the currently referenced module.

If the enumeration gives the set of j states such that $\{S_i \mid |E_i| = j$, all i's are different and $1 \leq i \leq k\}$. Then in the final states, all the S_i's are changed to m-1. The final state will be $(0, m-1, m-1, \ldots, m-1\ i-1, \ldots, m-1, \ldots, i_k)$

The transition probability is

$$f_h * \underbrace{(1 - 1/i_1) * \ldots * (1 - 1/i_k)}_{k-j} * \underbrace{(1/i_f) * \ldots * (1/i_r)}_{j}$$

c) States of the form $(0,\ldots,0,\ldots,i_1)$ $i_1 > 0$

The next states can be of the form:

i) $(0,\ldots,0)$, transition probability = $1/i_1$

ii) $(0,\ldots,0,i_1-1)$, transition probability = $(1-1/i_1)(1/p)$

iii) $(0,\ldots,0,p-1,i_1-1)$, transition probability = $(1-1/i_1)(1-1/p)$ (

iv) $(p-1,p-1,\ldots,p-1,i_1-1)$,
transition probability = $(1-1/i_1)(1-1/p)^{r-1}(1/p)$

v) $(p-1,p-1,\ldots,p-1,i_1-1,p-1)$fail,
transition probability = $(1-1/i_1)(1-1/p)^{r}$

d) States of the form $(0,\ldots,0,i_1, i_2, \ldots i_k)$ is similiar. The set of buffer contents which equal the currently referenced module must be enumerated.

e) States of the form $(i_1, i_2, \ldots i_r)$ where $i_j > 1$, $1 \leq j \leq r$. The set of states is enumerated similarily.

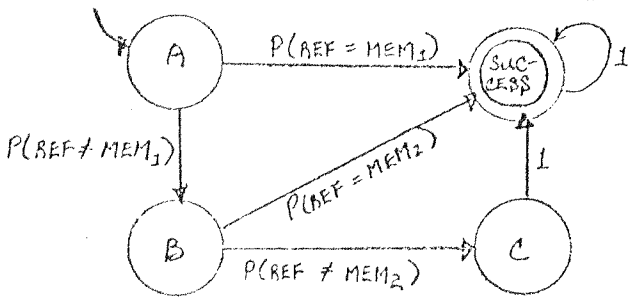
GENERATION OF TRANSITION MATRIX FOR P_2

Examples will be shown for the cases a) r=0, m=4; b) r=1, m=4; c) r=2, n=4. States lumping will also be investigated. A general algorithm for the generation of the transition matrix P_2 will be discussed.

Example (a)

r=0, m=4.

The state diagram can be represented as:



m_1, m_2 are consecutive memory modules.

The number of states is a function of the number of ways of interleaving. The transition matrix is:

$$P_2 = \begin{array}{c} \\ (A) \\ (B) \\ (C) \\ \text{Success} \end{array} \begin{array}{c} \begin{array}{cccc} (A) & (B) & (C) & \text{Success} \end{array} \\ \begin{bmatrix} 0 & 2/3 & 0 & 1/3 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$N = (I - Q)^{-1} = \begin{bmatrix} 1 & 2/3 & 1/3 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\pi' = [1 \quad 0 \quad 0]$$

$$\tau = N\xi = \begin{bmatrix} 2 \\ 2/3 \\ 1 \end{bmatrix}$$

$$\{M\pi'[t]\} = \pi'\tau = 2$$

$$\{\text{Var } \pi'[t]\} = \pi'(2N-I)\tau - (\pi'\tau)^2 = 2/3$$

For this case, $\#(S_1)$ (as calculated before) is $\frac{m}{m-1} = \frac{4}{3}$.

Therefore $F(0,4) = \frac{4/3}{(4/3)+2} = \frac{2}{5}$.

$$IMP_{max}(4) = \frac{1}{F(0,4)} = \frac{1}{2/5} = 5/2 = 2.5.$$

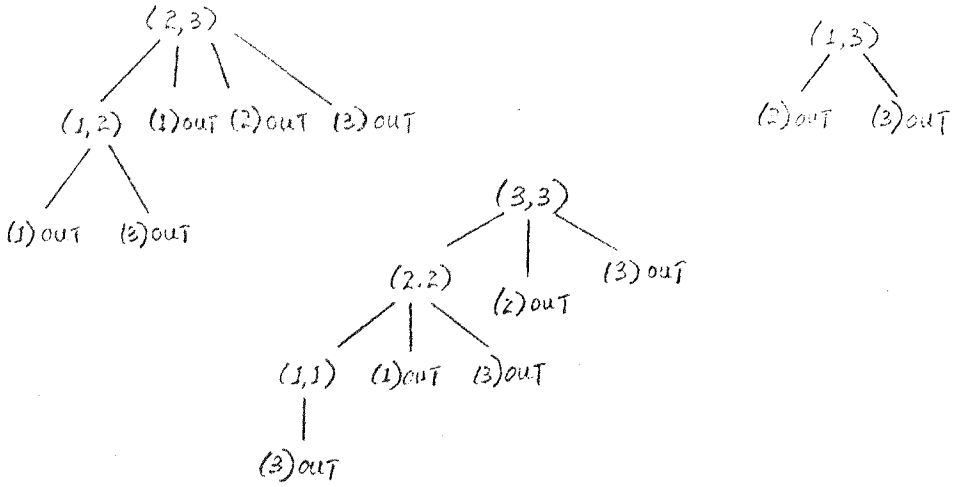
A computer solution has shown that for m up to 32, $IMP_{max}(m)$ satisfies the equation. $IMP_{max}(m) = 0.5(m+1)$

Example (b)

r=1, m=4

There are altogether 7 states (1,1) (1,2) (1,3) (2,2) (2,3) (3,3) success. The process can only start at initial states of (1,3) (2,3) and (3,3). This is because the last state variable in the initial states must be 3. The last state variable indicates the state of the memory reference in the input request stream when the process fails in S_1.

The state diagram can be drawn as:



initial state	final state	Explanation	Probability
(1,1)	(3) out	buf=mem ref=mem	1
(1,2)	(1) out	buf=mem ref≠mem	1/2
(1,2)	(3) out	buf=mem ref=mem	1/2
(1,3)	(2) out	buf=mem ref≠mem	2/3
(1,3)	(3) out	buf=mem ref=mem	1/3
(2,2)	(1,1)	buf≠mem ref≠mem	(1/2)(1/2)=1/4
(2,2)	(1) out	buf≠mem ref=mem or buf=mem ref≠mem	(1/2)(1/2)+(1/2)(1/2) = 1/4
(2,2)	(3) out	buf=mem ref=mem	(1/2)(1/2) = 1/4
(2,3)	(1,2)	buf≠mem ref≠mem	(1/2)(2/3) = 1/3
(2,3)	(1) out	buf≠mem ref=mem	(1/2)(1/3) = 1/6
(2,3)	(2) out	buf=mem ref≠mem	(1/2)(2/3) = 1/3
(2,3)	(3) out	buf=mem ref=mem	(1/2)(1/3) = 1/6
(3,3)	(2,2)	buf≠mem ref≠mem	(2/3)(2/3) = 4/9
(3,3)	(2) out	buf=mem ref≠mem or buf≠mem ref=mem	(1/3)(2/3)+(2/3)(1/3) = 4/9
(3,3)	(3) out	buf=mem ref=mem	(1/3)(1/3) = 1/9

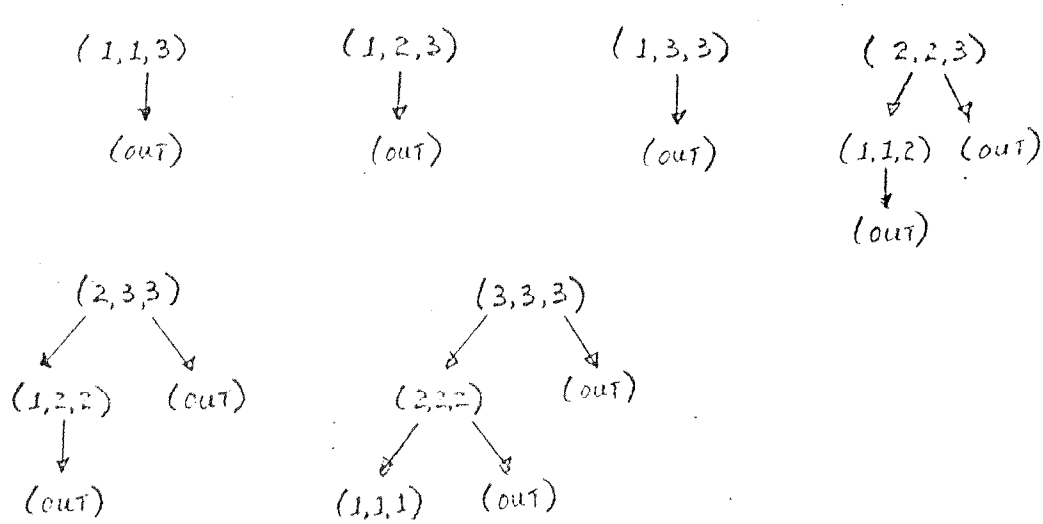
The transition matrix can be put together as:

	success	(1,1)	(1,2)	(1,3)	(2,2)	(2,3)	(3,3)
success	1	0	0	0	0	0	0
(1,1)	1	0	0	0	0	0	0
(1,2)	1	0	0	0	0	0	0
(1,3)	1	0	0	0	0	0	0
(2,2)	3/4	1/4	0	0	0	0	0
(2,3)	2/3	0	1/3	0	0	0	0
(3,3)	5/9	0	0	0	4/9	0	0

Example (c)

r=2, m=4.

There are 11 states. The process can start at initial states (1,2,3) (1,2,3) (1,3,3) (2,2,3) (2,3,3) and (3,3,3). To simplify the state diagram, all the success states are lumped into one, that is, we do not try to distinguish success states like (1,1) out, (2,2) out etc. The state diagram is represented as:



The transition matrix is

	(111)	(112)	(113)	(122)	(123)	(133)	(222)	(223)	(233)	(333)	succe
(111)	0	0	0	0	0	0	0	0	0	0	1
(112)	0	0	0	0	0	0	0	0	0	0	1
(113)	0	0	0	0	0	0	0	0	0	0	1
(122)	0	0	0	0	0	0	0	0	0	0	1
(123)	0	0	0	0	0	0	0	0	0	0	1
(133)	0	0	0	0	0	0	0	0	0	0	1
(222)	1/3	0	0	0	0	0	0	0	0	0	7/3
(223)	0	1/6	0	0	0	0	0	0	0	0	5/6
(233)	0	0	0	2/9	0	0	0	0	0	0	7/9
(333)	0	0	0	0	0	0	8/27	0	0	0	19/2
success	0	0	0	0	0	0	0	0	0	0	1

The different types of success states is not important in the generation of the transition matrix. However, it is important in the generation of the initial vector which will be considered later.

Reduction of the number of states by lumping

The criteria for lumping states has been considered before. Applying this to the transition matrix for the case r=2, m=4, we see that the states (1, 1, 3), (1, 2, 3) and (1, 3, 3) can be lumped together. This is because these states always succeeds in going to the success state and also one of its state variable is 3 which means that no other states can transfer to this state in S_2 (they are initial states only). The reduction does not have significant effects because the number of states in S_2 is much larger than the number of states in S_1 when r is large.

States lumping can only be applied when the number of state variables is greater or equal to 3. Since the first state variable must be a 1 and the last state variable must be a 3, the amount of reducation therefore equals the number of states in S_2 for r-2. The amount of states reduction and the number of resulting states for S_2 is shown in Fig. 5.

Fig. 5 Table of amount of states reduction and the resulting number of states in S_2

mem	4						8						16			
reg	2	3	4	5	6	7	2	3	4	5	6	7	2	3	4	5
reduction	3	6	10	15	21	28	7	28	84	210	462	924	15	120	680	3060
resulting no. of states	8	10	12	14	16	18	78	183	379	715	1255	2080	665	2941	10949	35701

Algorithm to Generate the Transition Matrix for S_2

The transition matrix for S_2 can be generated as follows:

1. For any states which contain a state variable equal to one, has a transition to the success state with a probability of 1.
2. For all other states, $(i_1, i_2, \ldots, i_{r+1})$ where $i_{r+1} \; i_r \; \ldots \; i_2 \; i_1 \; 1$, it has a transition to state $(i_1-1, i_2-1, \ldots, i_{r+1}-1)$ with a probability of $p = (1 - 1/i_1)(1 - 1/i_2)\ldots(1 - 1/i_{r+1})$ and a transition to the success state with a probability of 1-p. State lumping can be applied as the transition matrix is generated.

Generation of the Initial Probability Vector

Let IP_1 and IP_2 be the initial probability vector in S_1 and S_2 respectively. The process normally starts in state (0,0...0) in S_1 with an initial probability vector of (1,0,...,0). We can then determine the probabilities that it will go to the different failure states. These failure states become the initial states of S_2 and the grouping of these probabilities become an initial probability vector for S_2. This is repeated in S_2, and a set of probabilities is calculated for initial states in S_2 which will go to the different success states. These success states become the initial states for S_1. The process of successive approximation is repeated until the initial probabilities attain a steady state value. This process of successive approximation is time consuming, and therefore an initial probability vector is calculated from the basis of the transition that is most likely to be the success/

failure state. The transition probabilities from different "true" states to the same success/failure state are summed. The resulting vector of probabilities are then normalized to 1. The calculation is shown in the following 2 examples.

Example 1

This is to calculate the initial probability vector in S_2 for the case r = 1, m = 4.

From example (b) for the generation of the transition matrix in S_1, we get the following transitions to the failure state.

(0) --> (3,3) fail. p = 9/16

(3) --> (2,3) fail. p = 1/2

(2) --> (1,3) fail. p = 3/8

Grouping these probabilities into IP_2 vector, we get

(1,1)	(1,2)	(1,3)	(2,2)	(2,3)	(3,3)	success
0	0	3/8	0	1/2	9/16	0

Normalising it, we get

(1,1)	(1,2)	(1,3)	(2,2)	(2,3)	(3,3)	success
0	0	0.26	0	0.35	0.39	0

Example 2

This is to calculate the initial probability vector in S_1 for the case r=1, m=4

The following transition occurs in S_2 to the success state

(1,1) --> (1) out p = 1

(1,2) --> (1) out p = 1/2

(1,2) --> (3) out p = 1/2

(1,3) --> (2) out p = 2/3

(1,3) --> (3) out p = 1/3

(2,2) --> (1) out p = 1/2

(2,2) --> (3) out p = 1/4

(2,3) --> (1) out p = 1/6

(2,3) --> (2) out p = 1/3

(2,3) --> (3) out p = 1/2

(3,3) --> (2) out p = 4/9

(3,3) --> (3) out p = 1/9

Grouping these probabilities into IP_1 get

(0)	(1)	(2)	(3)	fail
0	7/6	10/9	97/36	0

Normalizing it, we get

(0)	(1)	(2)	(3)	fail
0	0.31	0.20	0.49	0

APPROXIMATION TECHNIQUE IN SOLVING $\pi(S_1)$

The number of states in the transition matrix of S_1 is very large, therefore it is necessary to investigate the approximate technique so that a lower bound can be found (the upper bound for the efficiency has been found before).

The transition matrices of S_1 for the cases of r=2, m=4 and r=3, m=4 are shown in Fig. 9 and Fig. 10. Both these matrices has been shown before. The second matrix has been shown in a slightly different way than before where the rows and columns are defined by a mapping function. Both matrices in here are shown with rows and columns in ascending order of the state variables.

By subdividing the two matrices into four regions according to the first state variable, we see that in regions II-IV, there is no transition back to the region itself and all transitions are directed to regions of lower order, especially to region I. Further, the probabilities for the transitions from region I to other regions are very small. Therefore little error will be incurred if we lump states which has the first state variable non-zero together. Fig. 11 shows the number of states resulted by such reduction.

The resulting amount of reductions is significant. However, sometimes the number of states is so large that by lumping states with the first state variable non-zero is not enough. The second row in Fig. 11 shows some of the reductions by lumping states with the first two state variables non-zero. The error incurred by lumping more states together will be bigger. For those combinations of m and r such that the resulting number of states are already small, no reduction is necessary. Alternatively, the matrix size can be fixed, and states exceeding that size are lumped. The second scheme will have a higher accuracy than the first. The exact difference will not be investigated.

Fig. 11 Table to Show the Resulting Number of States found by Approximation of Transition Matrix of S_1.

X represent that reduction is not possible for this combination of m and r.

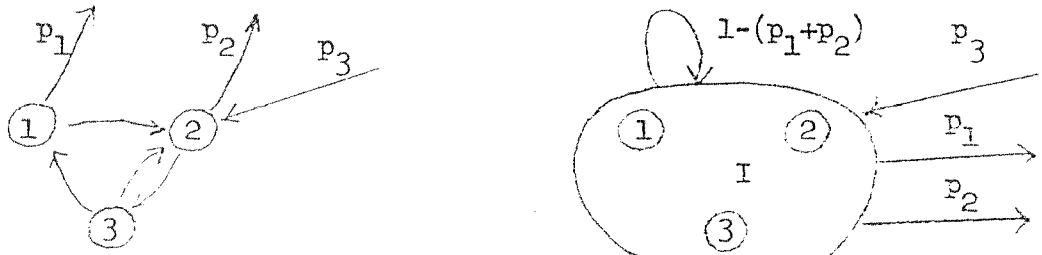
m	3																					
r	2	3	4	5	6	7	2	3	4	5	6	7	2	3	4	5	2	3	4	2	3	2	3	2	2
resulting no. of states by lumping states with 1st state variable ≠ 0	3	4	5	6	7	8	6	10	15	21	28	36	10	20	35	56	15	35	70	21	56	28	84	30	45	1
resulting no. of states by lumping states with 1st and 2nd state variable≠0	X	X	X	X	X	X	X	5	4	X	X	X	X	14	19	20	X	28	48	X	48	X	75	X	X	

To generate a consistent model when states are lumped, we will let $P_t(i/j)$ be the probability of going from j to i. Suppose that we want to lump the set of states $X = \{1, 2, 3 \ldots i\}$ together into a single state I, we will have in state I three different types of transitions:

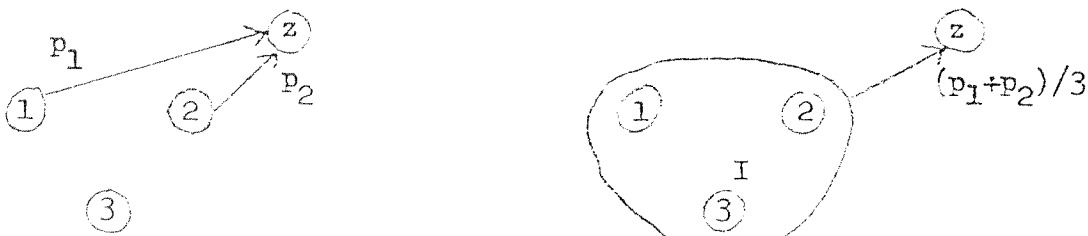
1. Those transitions $x \rightarrow y$ where $x, y \in X$ will be lumped into a single transition $I \rightarrow I$ with probability equals 1 - probability of all outgoing transitions (Fig. 12a). We will also assume that for any of the transitions going to I, it is equally likely to be in any of the states in X, i.e. $P_t(I/1) = P_t(I/2) = \ldots = P_t(I/i) = 1/i$.
2. Those transitions $z \rightarrow x$ where $z \notin X$ and $x \in X$ will be lumped into a single transition $x \rightarrow I$ by adding the transition probabilities, i.e. $P_t(I/z) = P_t(1/z) + P_t(2/z) + \ldots + P_t(i/z)$. (Fig. 12b)
3. Those transitions $x \rightarrow z$ where $z \notin X$ and $x \in X$ will be lumped into a single transition $I \rightarrow z$ by averaging the transition probabilities, i.e. $P_t(z/i) = P_t(I/1)P_t(1/z) + P_t(I/2)P_t(2/z) + \ldots + P_t(I/i)P_t(i/z)$. But since $P_t(I/1) = P_t(I/2) = \ldots = P_t(I/i) = 1/i$, we have $P_t(z/I) = (P_t(1/z) + P_t(2/z) + \ldots P_t(i/z))/i$. (Fig. 12c)

Fig. 12 Strategies in lumping states of S_t

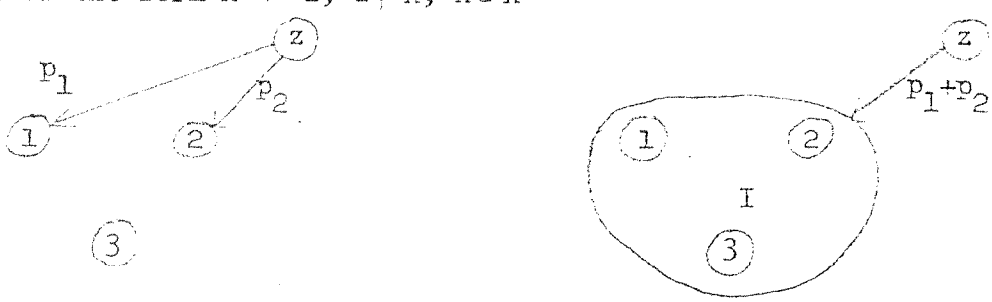
a. Transitions of the form $x \rightarrow y$, $x, y \in X$



b. Transitions of the form $z \rightarrow x$, $z \notin X$, $x \in X$



c. Transitions of the form $x \rightarrow z$, $z \notin X$, $x \in X$



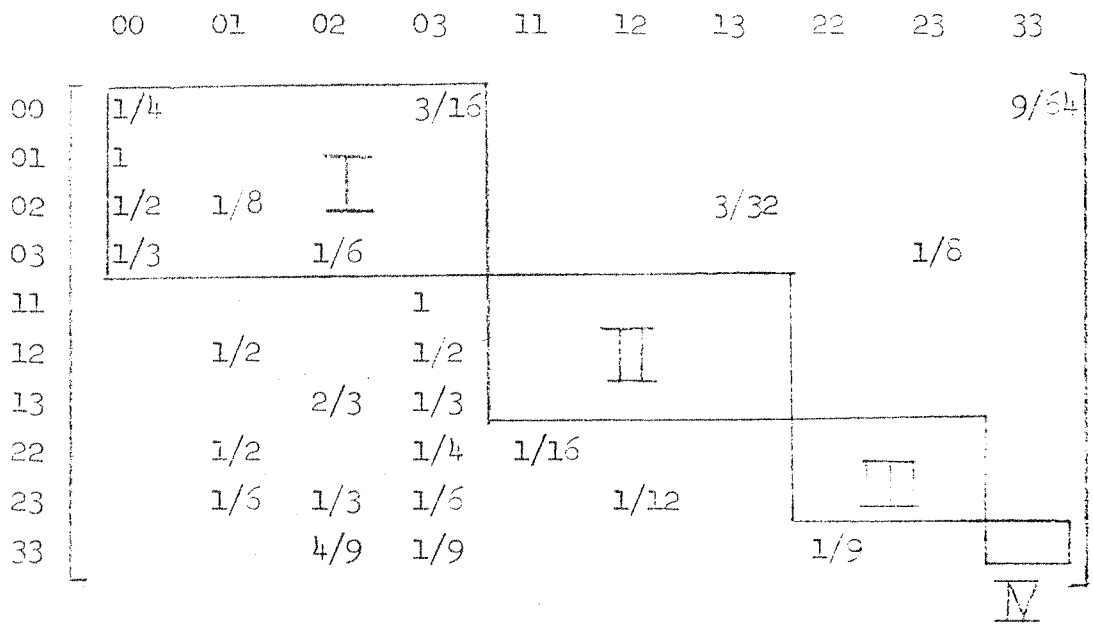


Fig. 9 Transition matrix for $S_{\underline{4}}$, r = 2, m = 4

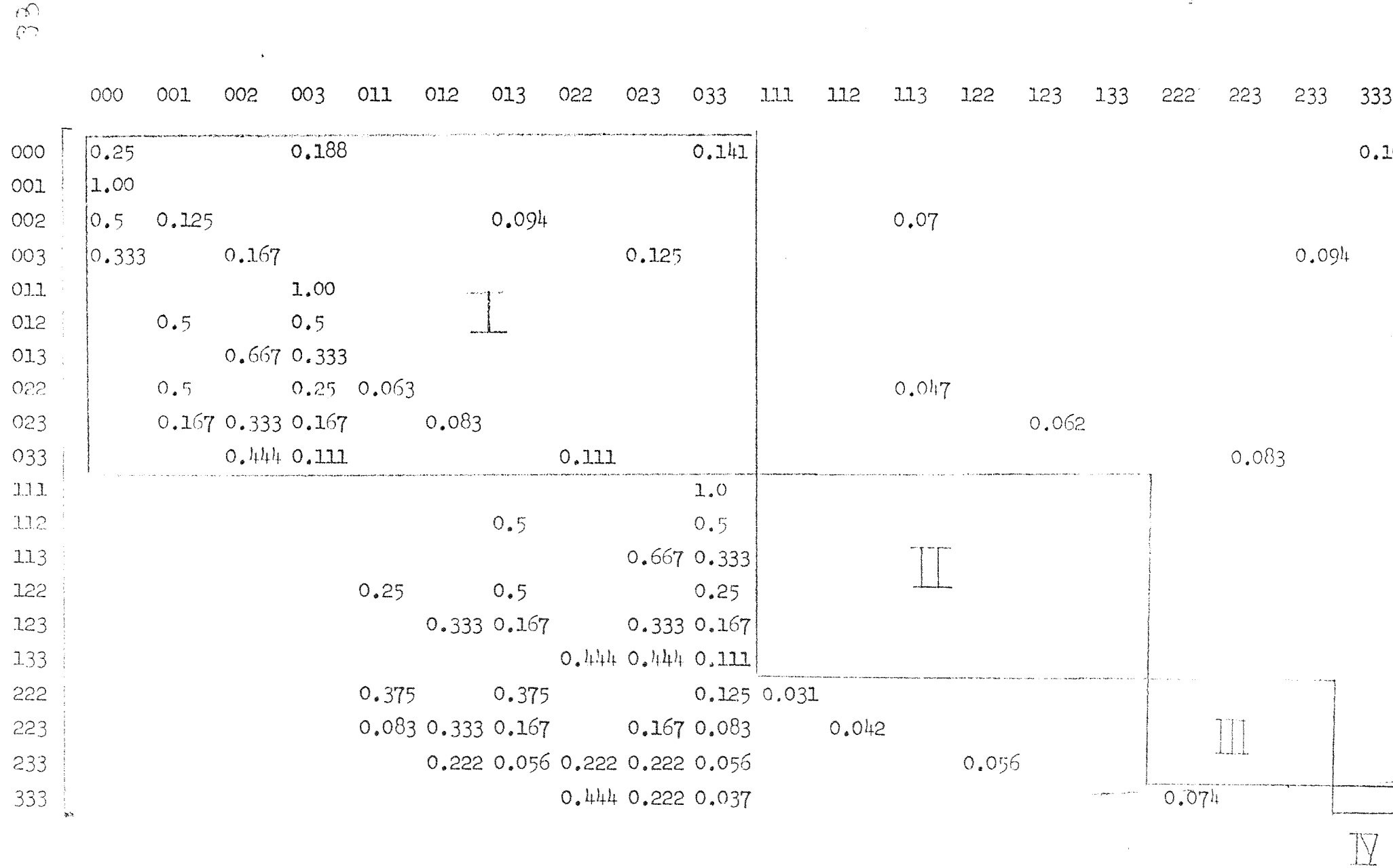


Fig. 10 Transition matrix for S_1, r = 3, m = 4.

SEGMENTATION TECHNIQUE IN SOLVING (S_2)

In the examples for the generation of the transition matrix for S_2, we see that the state diagrams are not "totally connected". There is a subset of states which can be the initial states and the next states that can go from these initial states are all disjoint (except the success state). Therefore it is more convenient to calculate the number of steps to success for each sub-state diagram and then find the weighed average of each. The maximum size of the transition matrix will be (m-1)(m-1) and the maximum number of iterations will be the total number of possible initial states. The number of iterations can be quite large in some cases. To reduce this, more than one initial states can be considered at one time. The size of the transition matrix generated by segmentation will be decreased exponentially (the size of the matrix equals the square of the number of states) and therefore the time to find the inverse will be smaller.

COMPUTER SOLUTION

Because of the large number of states, a computer solution is necessary. Fig. 6 shows the results of the computer solution. $s1 and $s2 are the number of states in s_1 and s_2. The matrix size is limited to 90 x 90. The choice of the value 90 is arbitary; it is limited by the central memory size of CDC6400 computer on which the analysis was done. Approximation was used when the matrix size was above 90. In the solution, there are only two instances where approximation is needed, namely, when reg = 5, 6 for mem = 5. The solution shows that an error of about 5 to 10 % is involved. Fig. 7 shows a graph of the relative improvement vs. the number of ways of interleaving. Fig. 8 shows a graph of the relative improvement vs. the number of register lookahead. The next section will try to extrapolate the results obtained and to find a closed form solution for the relative improvement.

REG	ME	$S1	$S2	MEAN(S1)	VAR(S1)	MEAN(S2)	VAR(S2)	F(R,P)	MAXIMP(P)	RELIMP(R,P)
1	2	2	1	6.000000	22.000000	1.000000	0.	.857143	1.500000	.571429
2	2	3	1	14.000000	142.000000	1.000000	0.	.933333	1.500000	.800000
3	2	4	1	30.000000	734.000000	1.000000	0.	.967742	1.500000	.903226
4	2	5	1	62.000000	3390.000000	1.000000	0.	.984127	1.500000	.952381
5	2	6	1	126.000000	14718.000000	1.000000	0.	.992126	1.500000	.976378
6	2	7	1	254.000000	61694.000000	1.000000	0.	.996078	1.500000	.988235
7	2	8	1	510.000000	253438.000000	1.000000	0.	.998043	1.500000	.994129
1	3	3	2	3.042857	4.265510	1.142857	1.000000	.726962	2.000000	.453925
2	3	6	3	5.535758	16.877139	1.054054	.594595	.840048	2.000000	.680098
3	3	10	4	9.450897	55.129991	1.022857	.434286	.902341	2.000000	.804682
4	3	15	5	15.438896	161.827330	1.010243	.358515	.938584	2.000000	.877168
5	3	21	6	24.521153	442.313217	1.004752	.318334	.960638	2.000000	.921276
6	3	28	7	38.212277	1148.627627	1.002254	.295274	.974442	2.000000	.948884
7	3	36	8	58.790508	2872.765722	1.001085	.281068	.983257	2.000000	.966514
1	4	4	3	2.259544	2.024645	1.333333	1.826087	.628895	2.500000	.381492
2	4	10	6	3.621951	6.196635	1.151174	1.081690	.758822	2.500000	.598036
3	4	20	10	5.569358	16.227544	1.077403	.772579	.837906	2.500000	.729843
4	4	35	15	8.261434	38.749823	1.042296	.615922	.887970	2.500000	.813284
5	4	56	21	11.948433	87.036376	1.024084	.527111	.921057	2.500000	.869429
6	4	84	28	16.949711	186.860317	1.014117	.472832	.943547	2.500000	.905911
7	4	87	36	36.558656	869.159445	1.008455	.437715	.973156	2.500000	.955260
1	5	5	4	1.924088	1.294313	1.533742	2.730061	.556444	3.000000	.334666
2	5	15	10	2.848065	3.422040	1.261303	1.569226	.693066	3.000000	.539600
3	5	35	20	4.086079	7.847610	1.145134	1.095265	.781096	3.000000	.671644
4	5	70	35	5.711813	16.506175	1.086247	.852963	.840212	3.000000	.760318
5	5	58	56	16.310297	137.473835	1.053472	.712796	.939329	3.000000	.908994
6	5	86	84	25.042644	375.581175	1.034140	.625069	.960343	3.000000	.940514
1	7	7	6	1.597455	.744919	1.941293	4.916261	.451418	4.000000	.268558
2	7	28	21	2.194588	1.747084	1.494368	2.650671	.594908	4.000000	.459877
3	7	84	56	2.899860	3.422320	1.296381	1.770717	.691061	4.000000	.588082
1	8	8	7	1.505897	.612164	2.145594	6.215912	.412406	4.500000	.244522
2	8	36	28	2.025449	1.411768	1.613213	3.261437	.556647	4.500000	.429074
1	9	9	8	1.441299	.530716	2.349688	7.656742	.380191	5.000000	.225239
2	9	45	36	1.902055	1.189823	1.732577	3.923174	.523314	5.000000	.404143
1	10	10	9	1.389293	.451533	2.553454	9.239576	.352367	5.500000	.208448
2	10	55	45	1.807598	1.031663	1.852140	4.637229	.493915	5.500000	.381451
1	16	16	15	1.229520	.251180	3.768487	21.726030	.246002	8.500000	.145469

FIG. 6. COMPUTER SOLUTION OF THEORETICAL ANALYSIS.

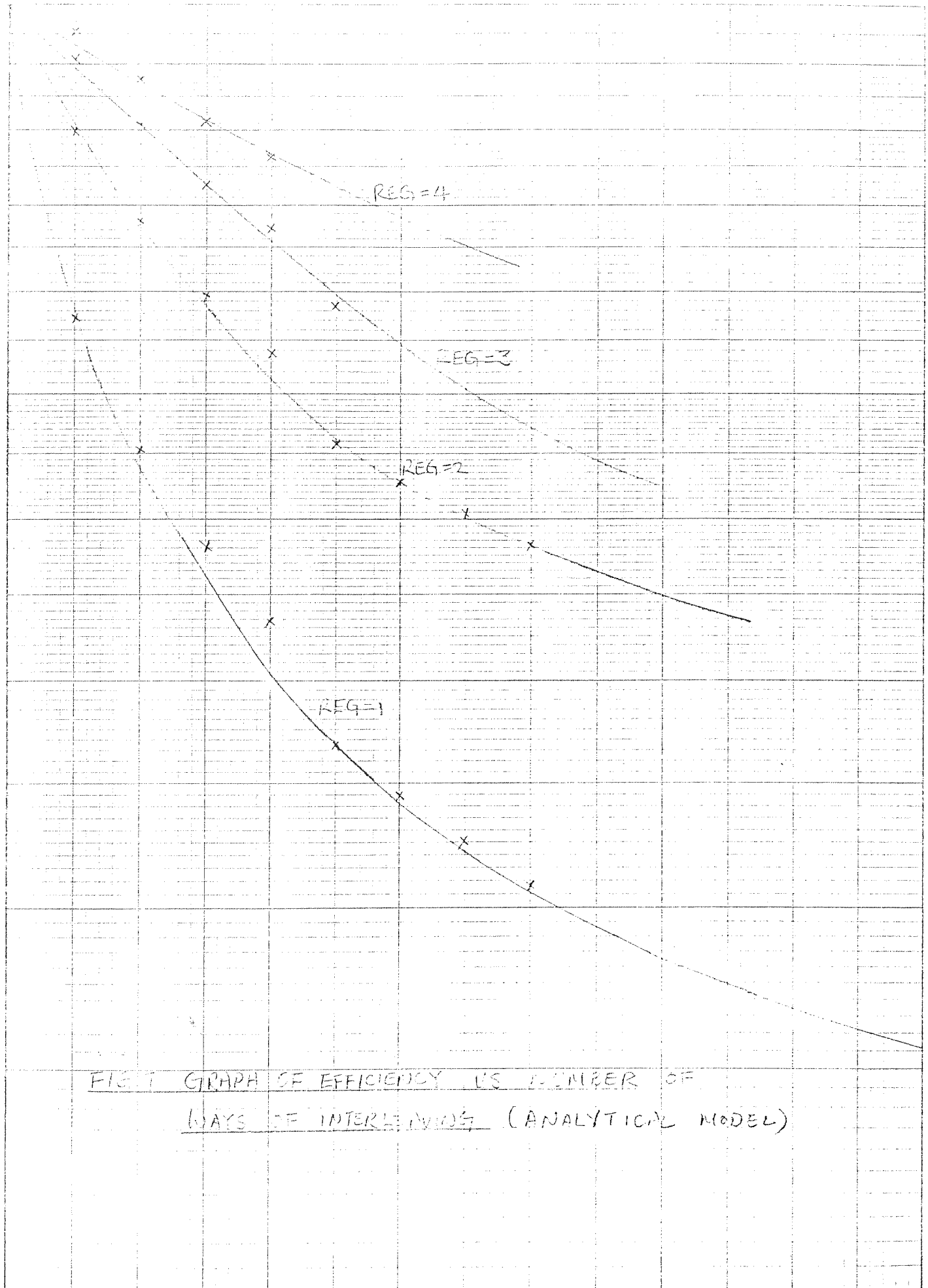


FIG 7 GRAPH OF EFFICIENCY VS NUMBER OF WAYS OF INTERLEAVING (ANALYTICAL MODEL)

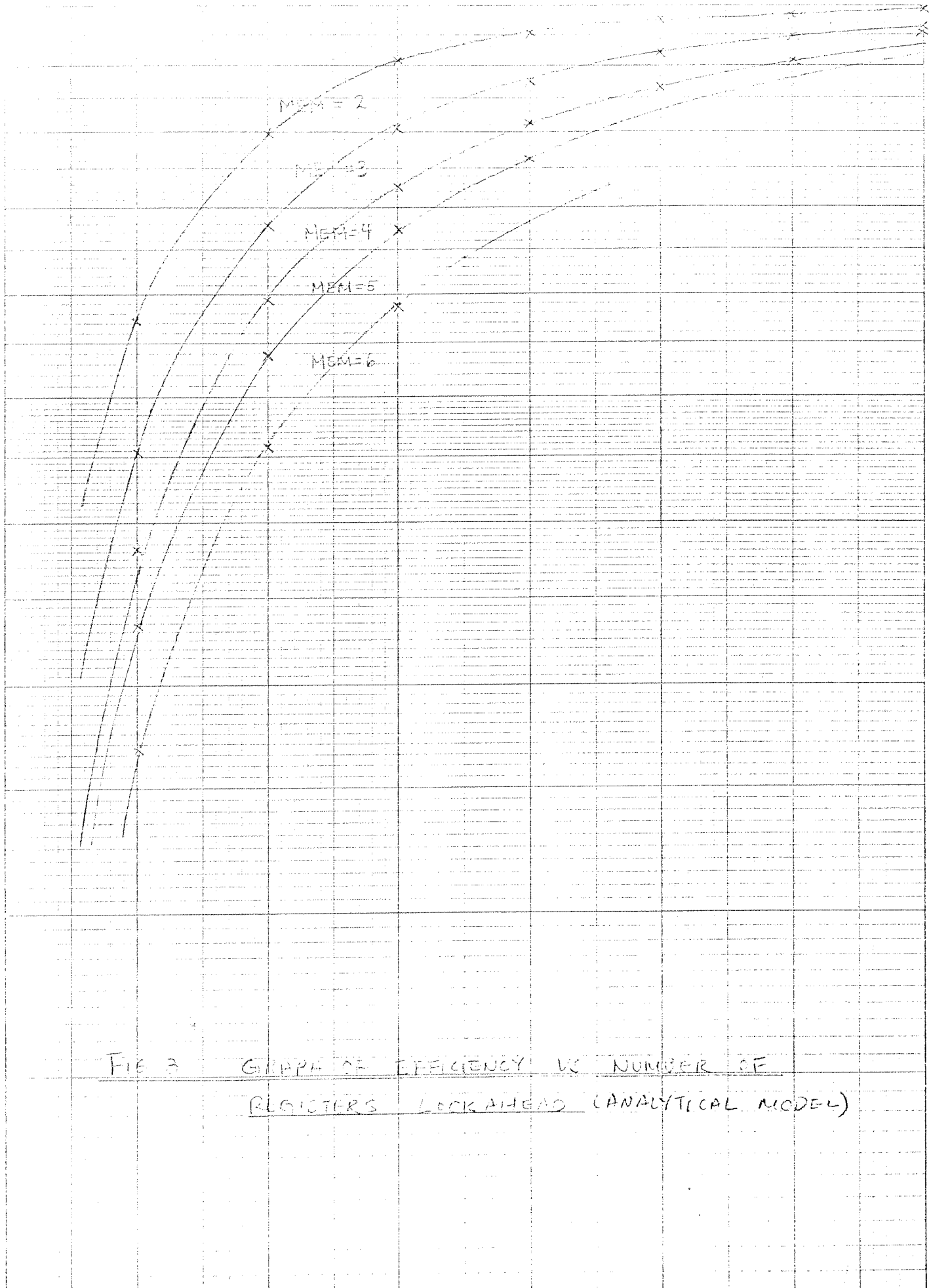


FIG 3 GRAPH OF EFFICIENCY VS NUMBER OF REGISTERS LOOKAHEAD (ANALYTICAL MODEL)

MATHEMATICAL EXTRAPOLATION FOR CONJECTURE SOLUTION

a) For the case m=2, assume that it has an exponential solution. Equation is of the form $EFF = 1 - e^{-br}$ where b is a positive constant.

Substituting

r=1:	$0.571429 = 1 - e^{-b}$	b = 0.84730
r=2:	$0.800000 = 1 - e^{-2b}$	b = 0.80472
r=3:	$0.903226 = 1 - e^{-3b}$	b = 0.77846
r=4:	$0.952381 = 1 - e^{-4b}$	b = 0.76113
r=5:	$0.976378 = 1 - e^{-5b}$	b = 0.74912
r=6:	$0.988235 = 1 - e^{-6b}$	b = 0.74044
r=7:	$0.994129 = 1 - e^{-7b}$	b = 0.73396

b) For the case m=4, equation is of the form $EFF = 1 - e^{-br}$.

r=1:	$0.439498 = 1 - e^{-b}$	b = 0.57892
r=2:	$0.598036 = 1 - e^{-2b}$	b = 0.45570
r=3:	$0.729843 = 1 - e^{-3b}$	b = 0.43625
r=4:	$0.813284 = 1 - e^{-4b}$	b = 0.41954
r=5:	$0.868429 = 1 - e^{-5b}$	b = 0.40564

Assume that the curve has an equation of the form $EFF = 1 - e^{-rb}$, where: r is the number of register lookahead

b is a constant depending on the number of ways of interleaving.

The following table shows the value of b for different r and m.

m	r	b	Average b
2	1	0.85	
	2	0.80	
	3	0.78	
	4	0.76	0.77
	5	0.75	
	6	0.74	
	7	0.73	

m	r	b	Average b
3	1	0.63	
	2	0.57	
	3	0.54	
	4	0.52	0.54
	5	0.51	
	6	0.50	
	7	0.49	

m	r	b	Average b
4	1	0.52	
	2	0.46	
	3	0.44	0.46
	4	0.42	
	5	0.41	
5	1	0.52	
	2	0.39	0.43
	3	0.37	

m	r	b	Average b
6	1	0.50	0.42
	2	0.34	
7	1	0.46	0.38
	2	0.31	
8	1	0.44	
9	1	0.41	
10	1	0.40	

Because of the lack of data, the value of b when $m > 7$ cannot be determined.

By using Newton's Interpolation Equation with forward differences,

$$f(m) = 0.77 - 0.23(m-2) + \frac{0.15}{2!}(m-2)(m-3) + \frac{0.10}{3!}(m-2)(m-3)(m-4) + \frac{0.07}{4!}(m-2)(m-3)(m-4)(m-5) + O((m-2)(m-3)(m-4)(m-5)(m-6))$$

Extrapolation can be done, however, when $m \gg 8$, the answers obtained will not be accurate.

$$\boxed{EFF = 1 - e^{-f(m)r}}$$

TRACE DRIVEN SIMULATION

As a general rule of thumb, the less the behavior of a system is known, the more likely simulation is needed. The more complicated a system is, the more "microscopic" the simulation must be. A further advantage in simulation is to verify our theoretical results. Our assumption in deriving the theoretical Markov model is that the memory accesses are random. On the other hand, memory accesses in real programs are more sequential in nature. Therefore, we can expect a better result in simulation.

A set of 17,000 traces was used to run the simulation. The trace was gotten from a CDC7600 Fortran program. The distribution of the type of memory accesses in the trace is as follows:

Type of Memory Access	Number	% of total
Instruction fetch (new access)	4572	27.1
Instruction fetch (no access necessary, instructions fetched in previous access)	7839	46.4
Operand fetch	2981	17.6
Operand store	1510	8.9
Total	16902	100.0

Fig. 9 shows the results from such a simulation. Fig. 10 shows a graph of the relative imporvement vs. the number of ways of memory interleaving. Fig. 11 shows a similiar graph of the relative improvement vs. the number of register lookahead. If these two graphs are compared with Fig. 7 and Fig. 8, a significant improvement is seen.

Some of the curves in Fig. 10 and Fig. 11 do not represent actual values. This is because approximation is applied at these points. Therefore the curves are not as smooth as what it should be.

Note: It takes an average of 75 seconds CPU time on a CDC6400 to simulate for one set of paramenters (register-memory pair) using a set of 17,000 traces.

2 FILES TO BE PROCCESSED

REG	MEM	MODE	$S1	$S2	PTS	MEAN(S1)	VAR(S1)	MEAN(S2)	VAR(S2)	F(R,P)	MAXIMP(P)	MINIMP(R,P)
1	9	3	10	9	16902	4.502306	30.200989	2.589813	9.922263	.634832	3.000000	.5[illegible]
2	9	3	46	37	16902	6.003389	57.853940	3.673899	15.673038	.620359	5.000000	.5[illegible]
1	10	3	11	10	16902	4.185064	25.603636	2.857814	12.181066	.594226	5.500000	.5[illegible]
1	16	3	17	16	16902	4.010223	22.692934	4.357179	31.344162	.479267	8.500000	.40[illegible]
THE COMBINATION OF				2		16 CANNOT BE REDUCED TO THE SIZE OF THE ARRAY SPECIFIED						
1	2	3	3	2	16902						1.500000	1.0[illegible]
2	2	3	4	2	16902						1.500000	1.0[illegible]
3	2	3	5	2	16902						1.500000	1.0[illegible]
1	3	3	4	3	16902	10.224471	174.623593	1.076378	.422115	.904753	2.000000	.809505
2	3	3	7	4	16902	87.561069	15831.196467	1.000000	0.	.988708	2.000000	.977417
2	4	3	11	7	16902	14.783658	472.150826	1.412720	1.747911	.912776	2.500000	.854626
3	4	3	21	11	16902	43.867176	8544.637783	1.904673	3.953271	.958388	2.800000	.930646
4	4	3	36	16	16902	60.221831	28190.525274	2.061538	4.646154	.966901	2.500000	.944834
1	5	3	6	5	16902	5.517464	47.239827	1.593577	3.030352	.775901	3.000000	.663852
2	5	3	16	11	16902	8.169816	118.310192	1.778982	3.224551	.821186	3.000000	.731779
3	5	3	36	21	16902	14.794092	550.374674	1.796788	3.521273	.891593	3.000000	.837389
4	5	3	71	36	16902	25.232181	2056.333723	1.914269	4.211826	.929484	3.000000	.894225
5	5	3	58	57	9608	96.851553	18643.195304	1.640870	2.761515	.983340	3.000000	.978010
3	3	3	11	5	16902						2.000000	1.000000
5	4	3	57	22	16902	71.804819	34002.959191	1.902141	3.954128	.974193	2.500000	.956988
6	5	3	86	85	16902	51.813171	5291.654328	1.977077	4.458072	.963245	3.000000	.944867
1	7	3	8	7	16902	4.927959	35.640718	2.118880	5.200922	.699315	4.000000	.599066
2	7	3	29	22	16902	6.883461	77.579547	2.807690	8.984477	.710282	4.000000	.613710
3	7	3	85	57	16902	10.047385	203.261950	2.448951	7.012976	.804026	4.000000	.736702
THE COMBINATION OF				4		7 CANNOT BE REDUCED TO THE SIZE OF THE ARRAY SPECIFIED						
1	8	3	9	8	16902	4.572931	30.812879	2.322677	7.709008	.663166	4.500000	.566927
2	8	3	37	29	16902	6.045107	59.185687	3.196630	11.726911	.654109	4.500000	.555284
3	8	3	86	85	16902	7.522376	98.164736	2.708368	8.446076	.735272	4.500000	.659635
1	9	3	10	9	5587	4.856867	34.454367	2.676193	10.630293	.644740	5.000000	.555425
1	6	3	7	6	16902	5.212226	40.305700	1.898992	4.738338	.732958	3.500000	.622142
2	6	3	22	16	16902	7.403777	89.486977	2.254096	5.674357	.765812	3.500000	.672137
3	6	3	57	36	16902	13.624930	400.384715	1.947797	4.202260	.874114	3.500000	.823760
4	6	3	72	71	16902	23.861268	1097.077774	2.101249	5.201981	.919066	3.500000	.886692
THE COMBINATION OF				5		6 CANNOT BE REDUCED TO THE SIZE OF THE ARRAY SPECIFIED						
1	4	3	5	4	16902	6.798832	72.444449	1.330195	1.856184	.836365	2.500000	.727275
6	4	3	85	29	16902	109.420230	68134.706851	1.800000	3.479070	.983816	2.500000	.973026
5	5	3	86	85	16902	51.813171	5291.654328	1.977077	4.458072	.963245	3.000000	.944867
THE COMBINATION OF				3		9 CANNOT BE REDUCED TO THE SIZE OF THE ARRAY SPECIFIED						
2	10	3	56	46	16902	5.256518	43.869340	4.012550	18.688950	.567103	5.500000	.470904

FIG 9 TRACE DRIVEN SIMULATION ANALYSIS.

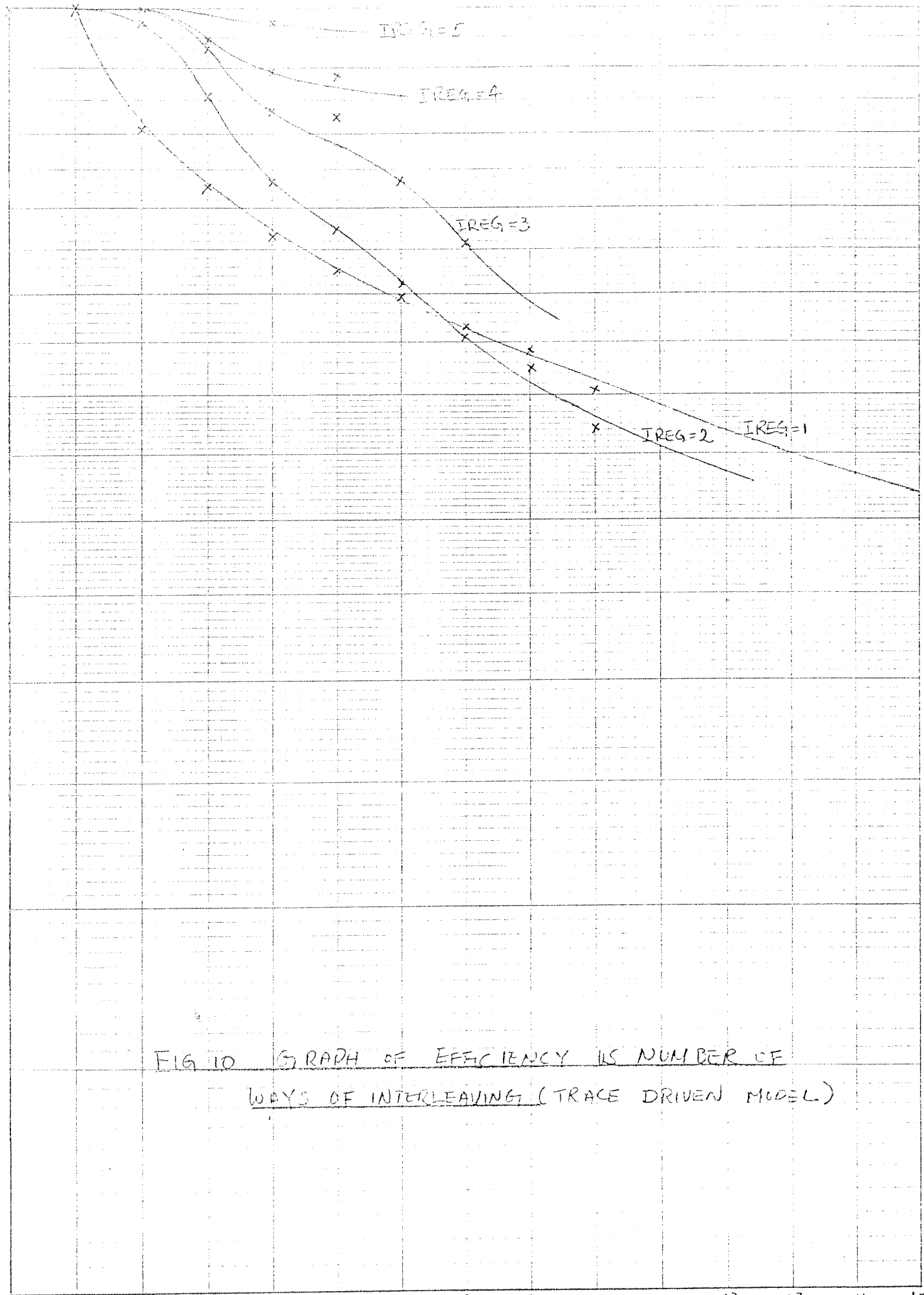


FIG 10 GRAPH OF EFFICIENCY VS NUMBER OF WAYS OF INTERLEAVING (TRACE DRIVEN MODEL)

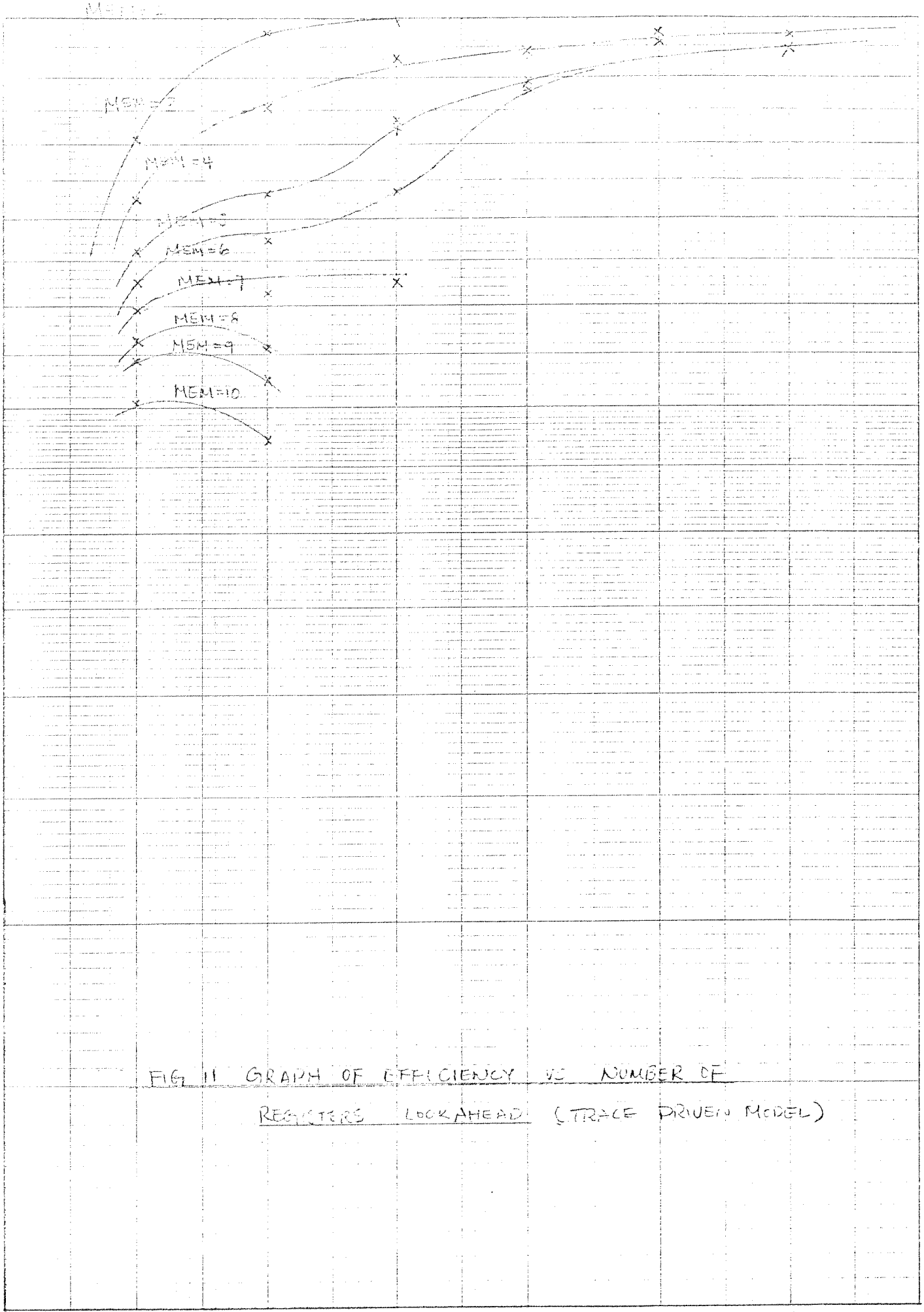


FIG 11 GRAPH OF EFFICIENCY VS NUMBER OF REGISTERS LOOKAHEAD (TRACE DRIVEN MODEL)

CONCLUSION AND EXTENSION

An intellegent buffering system for interleaved memory has been analyzed in this project. It is seen that for random requests, the improvement in performance is significant. For real programs, because of the sequential nature in memory accesses, the improvement is even greater. However, the number of states in a Markov chain solution is very large. The space and time constraints on a computer therefore prohibit a solution for some of the memory-register combinations. A closed form solution is also not possible in such an analysis.

In order to obtain a closed form solution for our system and also to confirm the analytical results, we can use queuing theory to obtain an alternate solution. The system can be viewed as a set of queues to each memory module. The sum of all the "customers" waiting in the queues must equal the number of buffers. There is also a request queue. The improvement can be found by assuming certain distributions for the arrival times and the service times.

Burnett and Coffman have proposed a memory system containing two separate queues, the instruction request queue and the data request queue. They showed that there is significant value of separately grouping instruction and data requests when accessing the memory. We can apply this approach to our model. Since data accesses are more random than instruction accesses, it would be beneficial to see the effect of increasing the amount of buffering for the data area. This is more cost effective than increasing

the degree of interleaving. Because program behavior is difficult to characterize, simulations will be the major tool we will use.

In order to fully understand the feasibility of our model, we must also look into the implementation aspect. The buffers are added in between the processor(s) and the memories. Therefore its design can be independent of both the configuration of the processor(s) and the memories. However, the speed must be compatible so that it will not introduce a bottle-neck. The buffers can be implemented with associative memory. Alternatively, this operation can be pipelined.

REFERENCES

(1) H. Hellerman, "Digital Computer System Principles" (Computer Science Series), 2nd Ed. New York: McGraw Hill, 1973, p245.

(2) D.E.Knuth, "Activity in an Interleaved Memory", IEEE Transactions on Computers, P 943-944, September 1975.

(3) D.E. Knuth, "The Art of Computer Programming, Vol. 1, Fundamental Algorithms, 2nd Ed. Reading, Mass. : Addison-Wesley, 1973, pp 112-119.

(4) C.V. Ravi, "On the Bandwidth and Interference in Interleaved Memory Systems.", IEEE Transactions on Computers, August 1972, pp 899-901.

(5) J.S. Litay, "Structural Aspects of the System 360/85, the Cache," IBM September, vol 7, no. 1, p 19, 1968.

(6) T.C. Chen, "Overlap and Pipeline Processing", Introduction to Computer Architecture, (ed. H.S. Stone), Science Research Associates, Inc., 1975, pp. 411-413.

(7) D.E. Knuth, "The Art of Computer Programming, Vol. 1, Fundamental Algorithms, 2nd Ed. Reading, Mass. : Addison-Wesley, 1973, Prob 6a) p.552.

(8) J.G. Kemeny, J.l Snell, "Finite Markov Chains," D. Van Nostrand Company, Inc, Princeton, New Jersey, 1960.

(9) A. Ralston, "A First Course in Numerical Analysis", McGraw Hill Book Co., New York, 1965.