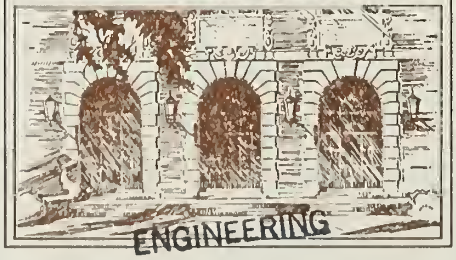


Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 5

ANALYSIS OF SIMULTANEOUS OPERATIONS AND MEMORY CONFLICT IN A MULTIMEMORY COMPUTER SYSTEM

by

Luther C. Abel

March 15, 1971

CAC Document No. 5

ANALYSIS OF SIMULTANEOUS OPERATIONS AND MEMORY CONFLICT IN A MULTIMEMORY COMPUTER SYSTEM

by

Luther C. Abel

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

March 15, 1971

ABSTRACT

This paper is an extension of the work of Shemer and Gupta [4] on the effect of multiple independent memory modules on processor efficiency and throughput. Their conclusions, although valid, were unnecessarily restricted. The results derived herein expand them to include instructions of differing execution times, instructions which do not require memory access, single as well as multiple instruction fetch and single as well as multiple word data fetches, plus the effects of a non-Poisson high-rate input-output device such as a swapping disk. An example of the application of the results to the ILLIAC IV computer system is given.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

This paper is an extension of the work of Shemer and Gupta [4] on the effect of multiple independent memory modules on processor efficiency and throughput. Their conclusions, although valid, were unnecessarily restricted. For example, their results were derived under the clearly unrealistic assumption that the execution times of all processor instructions were equal. The results derived herein expand Shemer and Gupta's results in that instructions of differing execution times, instructions which do not require memory access, single as well as multiple instruction fetch, and single as well as multiple word data fetches are permitted and the effects of a non-Poisson high-rate input-output device such as a swapping disk are included.

SYSTEM MODEL AND OPERATION

Consider the system configuration shown in Figure 1. The processor and input-output controller have independent data paths for communication with each of the N (the notation of Shemer and Gupta [4] will be used insofar as applicable) independent memory modules which form the primary memory of the system. In order to minimize data queue lengths in the input-output controller (typically only a one word queue is implemented) and to prevent obliteration of waiting data by other closely following data, memory cycle requests from the input-output controller are given non-preemptive priority over those from the processor.

M contiguous words are fetched simultaneously at any memory access. For $M > 1$, this corresponds to the concept of fetching an M-word superword. Addresses are interlaced among the independent memory banks such that address "a" is in memory bank (a div M mod N). The processor fetches M instructions simultaneously, and M' of these require an operand from memory. Thus each M instructions require $M' + 1$ memory cycles. It is assumed that all memory cycles require time C.

Let the instruction set of the processor consist of a repertoire of R instructions, $g_1, g_2, \ldots, g_R$ of length $r_1, r_2, \ldots, r_R$ respectively. We add to this instruction set the instruction fetch operation g_0, which requires time r_0 between initiation and execution of the first fetched instruction.

Some of the R instructions require memory accesses during their execution, others do not. Without loss of generality, let us order the elements of the instruction set such that the first $Q - 1$ instructions require a memory cycle. We then define

$$G' = \{g_i \mid 0 \leq i \leq Q - 1\}$$

$$G'' = \{g_i \mid Q \leq i \leq R\} .$$

To convert this instruction set $G = G' \cup G''$ into a memory-oriented set

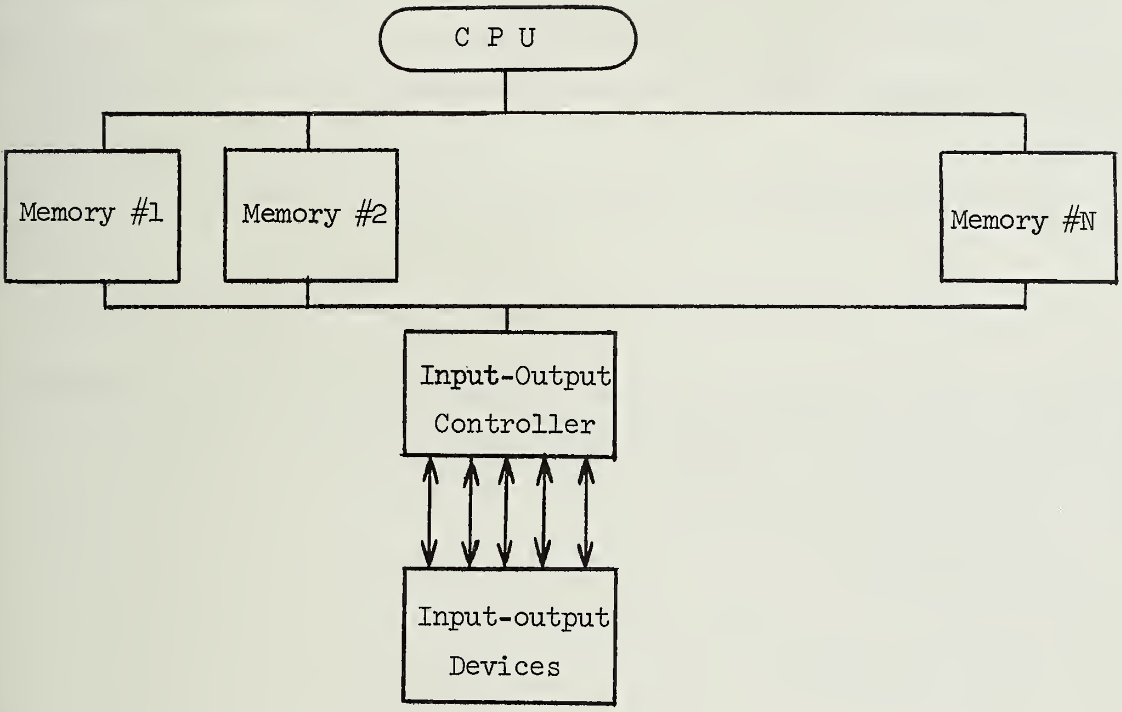


Figure 1. System Configuration

of operations, we define

$$h_i^k = \{(g_i, g_x, g_y, \ldots, g_z, G') \mid g_i \in G', h_x, h_y, \ldots h_z \in G''\}.$$

That is, h_i^k is a n-tuple consisting of the i^{th} processor instruction which requires a memory cycle, followed by some number of non-memory accessing instructions, and terminated by any instruction which does require a memory access. The superscript k is simply a member of the index set which allows us to record all possible sequences of the instruction g_i followed by some number of non-memory reference instructions before the next instruction requiring a memory access.

Associated with each h_i^i there is an execution time

$$t_i^k = r_i + r_x + r_y + \ldots + r_z$$

(note that it does not include the time to execute the terminating, memory-referencing instruction). Each h_i^k occurs with a certain probability, PR $[h_i^k]$, in an instruction stream.

We now define

$$h_i = \bigcup_k h_i^k .$$

The set

$$H = \{h_i\}$$

is then a memory-oriented pseudo-instruction set for the processor. Associated with each pseudo-instruction, h_i, is a probability

$$\Pr [h_i] = \sum_k \Pr [h_i^k] ,$$

and an average execution time

$$t_i = \sum_k \Pr [h_i^k] \; t_i^k .$$

ne average duration of a pseudo-instruction is the expectation of t_i ,

$$E[T] = \sum_{i=0}^{Q-1} \Pr[h_i]\, t_i \, .$$

A number of questions arise:
nat is the probability that a processor's memory access will be delayed
ecause of input-output traffic, as a function of that traffic density?
urther, what is the average duration of such a delay, and how does the per-
ormance of the likelihood that the processor will be delayed, either by it-
elf or by an input-output transaction? In the following section, the analysis
ill seek to answer these questions.

ANALYSIS

An exact, detailed analysis would be extremely difficult, requiring differing conditional relations for each step of a program describing the previous history of the program and its environment and the current state of all components of the system. The system is therefore examined on a statistical basis assuming: a) stationary conditions, b) generation of memory requests by the input-output controller is characterized by a combination of a Poisson process and a high speed device (such as a swapping disk) generating requests at a uniform rate, c) if the processor is denied a memory access during execution of an instruction, it will wait rather than proceed with any other task.

The input-output controller is assumed to be simultaneously regulating and providing memory service for a large number of peripherals such as tapes, printers, card equipment, and teletypes. Since memory requests arise from a large number of independent sources, the generation of memory commands by the input-output controller is considered to be partially governed by a Poisson phenomenon with cumulative request rate λ. It is assumed that the input-output controller also services a high speed device which generates memory access requests with uniform rate γ when actuated. This device has a probability of being actuated (i.e., a duty cycle) of Pr [X].

Since there are N independent memory banks and addresses are interlaced among them, it is reasonable to assume that the cumulative demands λ and γ are evenly divided among all N memories. Since each memory cycle is of length C, the input-output controller imposes a load

$$\rho = \frac{\lambda C}{N} + \frac{\Pr[X]\ \gamma C}{N}$$

$$= \frac{(\lambda + PR\ [X]\ \gamma)C}{N}$$

$$= \frac{\zeta C}{N} \qquad \text{where } \zeta = \lambda + PR\ [X]\ \gamma\ .$$

Therefore under stationary conditions with $\rho < 1$, the probability that k out of N memory modules are <u>not</u> busy is

$$\Pr[k] = \binom{N}{k} (1 - \rho)^k \rho^{N-k} .$$

We denote by "A" the event that the processor does not experience delay in memory access due to input-output servicing. Assuming independence between processor and input-output memory access addresses,

$$\Pr[A \mid k] = k/N.$$

Averaging over all values of k, we obtain

$$\Pr[A] = \sum_{k=1}^{N} \frac{k}{N} \binom{N}{k} (1 - \rho)^k \rho^{N-k} .$$

Closing the series,

$$\Pr[A] = (1 - \rho).$$

This should not be a surprising result, since if ρ is the probability that a memory bank will be busy servicing an input-output controller request then $1 - \rho$ should be the probability that the bank is available to the processor.

Even if the processor requests a memory cycle from a memory bank which is not servicing an input-output controller command, it is still possible for the servicing of the request to be delayed if the memory bank is still busy servicing a previous processor request.

Let

$$S^k = \{S_1^k, S_2^k, \ldots, S_Q^K\}$$

where S_j^k is the k-tuple of memory-oriented pseudo-instructions $(h_x, h_y, \ldots, h_z)$, $x, y, \ldots z \in \{0, 1, 2, \ldots, Q-1\}$, and j is the image of x, y, ..., z in a 1-1 mapping of k-tuples of non-negative integers less than Q onto the positive integers not greater than Q^k.

$$T_j^k = t_x + t_y + \ldots + t_z$$

is the time required to execute the pseudo-instruction sequence S_j^k if no delay occurs. The expectation $E[T^k]$ is the average value of T_j^k,

$$E[T^k] = \sum_{j=1}^{Q^k} \Pr[S_j^k]\, T_j^k \, .$$

Now consider the execution of a sequence of k memory-oriented pseudo-instructions. Let $\Pr[I_k']$ be the probability that the input-output controller requests access to a particular memory bank due to the high speed swapping device, and $\Pr[I_k'']$ the probability of an input-output generated cycle in that same memory due to any of the other input-out devices. Then

$$\Pr[I_k'] = \sum_{j=1}^{Q^k} \Pr[S_j^k] \Pr[X] \min\left(1, \frac{\gamma T_j^k}{N}\right)$$

and

$$\Pr[I_k''] = \sum_{j=1}^{Q^k} \Pr[S_j^k] \left(1 - \exp(-\lambda T_j^k/N)\right) \, .$$

These probabilities are of interest in the following work for small values of k only; contemporary primary and secondary memory speeds result in

$$\frac{\gamma T_j^k}{N} \leq 1$$

for small k. Therefore,

$$\Pr[I_k'] = \Pr[X] \sum_{j=1}^{Q^k} \Pr[S_j^k] \frac{\gamma}{N} T_j^k$$

$$= \Pr[X] \frac{\gamma}{N} E[T^k] \, .$$

The cumulative request rate λ is quite small also. Typically, each word transferred to a computer's memory by an input-output device (other than a swapping disk which has been considered separately) will require several processor cycles of manipulation. Therefore, $\lambda \ll 1$ and for small k the approximation

$$1 - \exp(-\lambda T_j^k/N) = \lambda T_j^k/N$$

can be used. Then

$$\Pr[I_k''] = \sum_{j=1}^{Q^k} \Pr[S_j^k] (\lambda/N) T_j^k$$

$$= (\lambda/N) E[T^k] .$$

Let K_k denote the event that no input-output controller memory access occurs in a particular memory bank during the execution of k pseudo-instructions by the processor. Then

$$\Pr[K_k] = 1 - \Pr[I_k'] - \Pr[I_k''] .$$

Again for small values of k which are of interest,

$$\Pr[K_k] = 1 - \Pr[X] (\gamma/N) E[T^k] - (\lambda/N) E[T^k]$$

$$= 1 - (E[T^k]/N) (\lambda + \gamma \Pr[X]) .$$

Using the previously defined $\zeta = \lambda + \gamma \Pr[X]$,

$$\Pr[K_k] = 1 - (\zeta/N) E[T^k] .$$

The probability of occurrence of one of the Q pseudo-operation codes in, e.g., the ith position of an instruction sequence of length k is certainly not uncorrelated with the occurrence of certain operations in the other k-1 positions. For example, a store operation is almost certainly followed by a load and was (particularly for compiler generated code) likely preceded by a series of arithmetic operations. Nevertheless, it is a reasonable first-order approximation to assume the occurrence of operation codes in various positions of the sequence to be uncorrelated with the occurrence of instructions in other positions. Then

$$E[T^k] = k E[T]$$

and

$$\Pr[K_k] = 1 - \frac{k\zeta}{N} E[T] .$$

If the processor is delayed because the ith preceding pseudo-instruction required access to the same memory bank, let this be denoted by B_i.

For example

$$\Pr[B_1] = (1/N) \Pr[A] \Pr[L_1] \Pr[K_1] ,$$

where $\Pr[L_1]$ is the probability that the previous pseudo-instruction was of length less than C. Similarly

$$\Pr[B_2] = (1/N) \Pr[A] \Pr[L_2] \Pr[A] \left(\frac{N-1}{N}\right) \Pr[K_2]$$

and in general

$$\Pr[B_k] = (1/N) \Pr[A] \Pr[L_k] \left(\Pr[A] \left(\frac{N-1}{N}\right)\right)^{k-1} \Pr[K_k] .$$

Thus the probability that the processor is undelayed by any previous processor memory access, denoted by the event B, is

$$\Pr[B] = \frac{\Pr[A]}{N} \sum_{k=1}^{\infty} \Pr[L_k] \Pr[K_k] \left(\Pr[A] \left(\frac{N-1}{N}\right)\right)^{k-1}$$

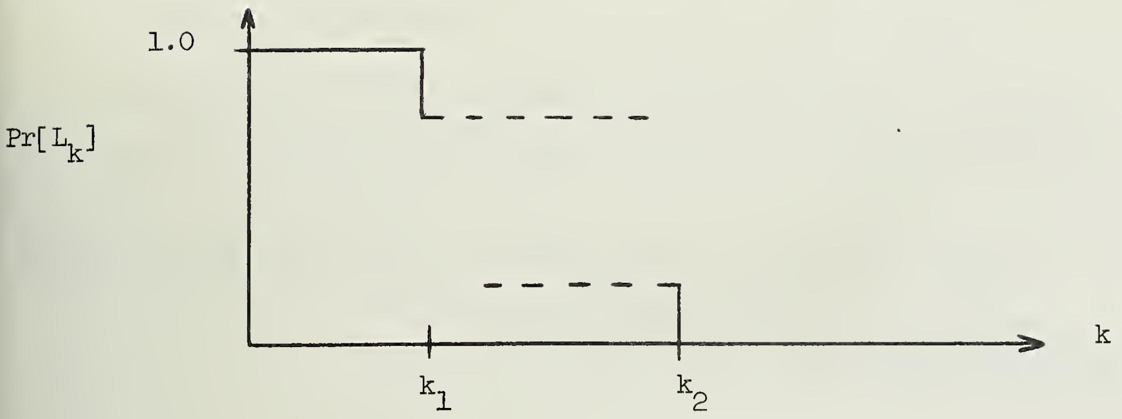
$$= \frac{1}{N-1} \sum_{k=1}^{\infty} \Pr[L_k] \Pr[K_k] \left(\Pr[A] \left(\frac{N-1}{N}\right)\right)^{k} .$$

The evaluation of $\Pr[L_k]$ is in general a difficult task since the probability curve has the appearance shown in Figure 2 (T_ℓ is the execution time of the longest pseudo-instruction, T_s the execution time of the shortest). One simplifying assumption is that $\Pr[L_k]$ in the region between k_1 and k_2 can be expressed as a simple linearly decreasing function

$$\Pr[L_k] = (k_2 - k) / (k_2 - k_1) , \quad k_1 \leq k \leq k_2 ,$$

perhaps with judicious juggling of the values of k_1 and k_2 if the instruction set includes rarely used, extremely short or long operation codes. With such an approximation, the series can be closed. The form of the closed expression is so complex, however, as to obscure simple observation of the effects of ζ, N, and processor speed on the likelihood of processor delay.

We note that a computer's instruction set invariably includes several instructions which require time considerably in excess of one memory cycle for execution. In that case $k_1 = 1$ and the $\Pr[L_k]$ curve may appear as in



where $k_1 = \lfloor C/T_\ell \rfloor$

$k_2 = \lceil C/T_s \rceil$

Figure 2. Generalized $Pr[L_k]$ Curve

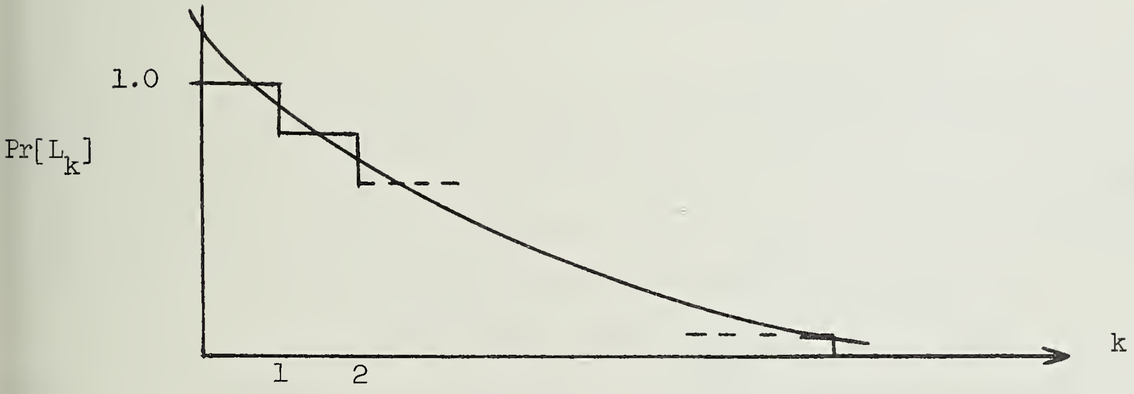


Figure 3. Typical $Pr[L_k]$ Curve With Exponential Approximation

Figure 3. We therefore make the alternative simplifying assumption that $\Pr[L_k]$ can be approximated by the exponential

$$\Pr[L_k] = \alpha_1 \exp(-\alpha_2 k)$$

where α_1 and α_2 are constants adjusted to best fit the actual $\Pr[L_k]$ curve.

Substituting for $\Pr[L_k]$ and $\Pr[K_k]$ in the expression for $\Pr[B]$,

$$\Pr[B] = \frac{1}{N-1} \sum_{k=1}^{\infty} \alpha_1 \exp(-\alpha_2 k) \left(1 - \frac{k\zeta}{N} E[T]\right)\left(\Pr[A] \frac{N-1}{N}\right)^k$$

Again, for small values of ζ,

$$1 - \frac{k\zeta}{N} E[T] = \exp(-k\zeta E[T]/N)$$

and

$$\Pr[B] = \frac{\alpha_1}{N-1} \sum_{k=1}^{\infty} \left(\exp(-\alpha_2 - \zeta E[T]/N) \Pr[A] \frac{N-1}{N}\right)$$

Define

$$\beta = \exp(-\alpha_2 - \zeta E[T]/N) \Pr[A] (N-1)/N .$$

Then

$$\Pr[B] = \frac{\alpha_1}{N-1} \sum_{k=1}^{\infty} \beta^k ,$$

and closing the series,

$$\Pr[B] = \frac{\alpha_1}{N-1} \frac{\beta}{1-\beta}$$

$$= \frac{(1-\rho)\left(\frac{\alpha_1}{N}\right) e^{-\alpha_2} e^{-\zeta E[T]/N}}{1 - (1-\rho)\left(\frac{N-1}{N}\right) e^{-\alpha_2} e^{-\zeta E[T]/N}} .$$

We note that for N = 1, this simply reduces to the expression for $\Pr[B_1]$

$$\Pr[B_1] = (1-\rho)\,\alpha_1 e^{-\alpha_2} e^{-\zeta E[T]/N} ,$$

which is simply the probability that the previous pseudo-instruction was less than C in length, diminished by the probability that an input-output transaction has occurred or is pending.

The probability that a processor is delayed due to any cause is

$$\Pr[D] = \Pr[A' \text{ or } B] = (1 - \Pr[A]) + \Pr[B]$$

where $\Pr[A'] = 1 - \Pr[A]$ is the probability that the processor is delayed due to input-output servicing.

$$\Pr[D] = \rho + \frac{(1-\rho)\,(\alpha_1/N)\,\exp(-\alpha_2)\,\exp(-\zeta E[T]/N)}{1 - (1-\rho)\,(\frac{N-1}{N})\,\exp(-\alpha_2)\,\exp(-\zeta E[T]/N)}$$

If event D occurs (i.e., the processor is delayed), it is natural to consider the length of the delay. First consider the delay if the interference is caused by the input-output controller. Let W_c be the length of the delay resulting from input-output controller access of a memory bank; define T_c to be the time remaining for completing an in-process memory cycle when the processor request occurs, and let $E[n_1]$ denote the expected number of input-output memory access requests queued for this memory. The mean value of W_c is then given by

$$E[W_c] = E[T_c] + E[n_1]\,C + \frac{\zeta C}{N} E[W_c] \quad .$$

The three terms of this expectation are the average amount of time needed to finish an in-process command, the average time to complete the remainder of the queued requests, and the average time required to service any new commands which arrive while the other requests are being serviced (these new requests will be serviced ahead of the processor's request).

To arrive at $E[W_c]$, $E[n_1]$ and $E[T_c]$ must be obtained. The average length of the queue, $E[n_1]$, is related to the average time an arriving request must wait in the queue, $E[W_1]$:

$$E[W_1] = E[T_c] + \frac{\zeta C}{N} E[W_1] \quad .$$

Here the first term is again the time required to complete an in-process command, and the second is the time needed to empty the queue. From this,

$$E\,[W_1] = \frac{E\,[T_c]}{1-\zeta C/N} = \frac{E\,[T_c]}{1-\rho} \quad .$$

Since the system is in equilibrium, the average size of the queue [3] is simply the number of additional requests which arrive while a request is waiting for service,

$$E\,[n_1] = \frac{\zeta}{N}\,E\,[W_1] = \frac{\zeta\,E\,[T_c]}{N\,(1-\rho)}$$

Substituting this into the equation for $E\,[W_c]$ and solving,

$$E\,[W_c] = E\,[T_c]\,/\,(1-\rho)^2 \quad .$$

The arrival input-output memory access requests are uncorrelated with respect to both other input-output requests and processor commands. Therefore the probability of the arrival of a request when a memory bank is busy may be assumed to be uniformly distributed over the memory cycle time C, whence the expected waiting time for access is simply

$$E\,[T_c] = C/2 \ .$$

Thus

$$E\,[W_c] = \frac{C}{2\,(1-\rho)^2} \quad .$$

Now consider the processor delay due to the kth previous pseudo-instruction being directed to the same memory bank. Let W_p^k be the duration of such a delay, and $E\,[T_p^k]$ be the average time to complete a previously initiated memory cycle. To this completion time we add the delay due to any input-output transactions arriving during the period:

$$E\,[W_p^k] = E\,[T_p^k] + (\zeta/N)\,(C - E\,[T_p^k] + E\,[W_p^k])\,C \quad ,$$

from which

$$E\,[W_p^k] = \frac{\rho C + (1-\rho)\,E\,[T_p^k]}{(1-\rho)}$$

$$= \rho C/(1-\rho) + E\,[T_p^k] \quad .$$

Given that the processor is delayed, the expected value of the waiting time is therefore

$$E[W \mid D] = \frac{\Pr[A'] \, E[W_c] + \sum_{k=1}^{\infty} \Pr[B_k] \, E[W_p^{\,k}]}{\Pr[D]}$$

The expected time to complete a memory cycle due to the kth previous pseudo-instruction is in general as intractable to analysis as the previous quantities we have used. However, it is not unreasonable to assume that this expectation is also exponentially related to k:

$$E[T_p^{\,k}] = C \exp(-\alpha_3 k) \quad .$$

Unfortunately, α_3 is not in general directly related to the α's used in the approximation to $\Pr[L_k]$, and hence must be separately determined. We define $\alpha_4 = \alpha_2 + \alpha_3$. Closing the series in the expression for $\Pr[W \mid D]$,

$$\sum_{k=1}^{\infty} \Pr[B_k] \left(\rho C/(1-\rho) + E[T_p^{\,k}]\right)$$

$$= \frac{C\,\alpha_1}{N-1} \sum_{k=1}^{\infty} \exp(-\alpha_2 - \zeta E[T]/N)\left(\Pr[A] \frac{(N-1)}{N(1-\rho)}\right)^k$$

$$+ \sum_{k=1}^{\infty} \exp(-\alpha_4 - \zeta E[T]/N)(\Pr[A]\,(N-1)/N\,)^k$$

$$= \frac{(\rho C\alpha_1/N)\exp(-\alpha_2 - \zeta E[T]/N)}{1 - (\frac{N-1}{N})\exp(-\alpha_2 - \zeta E[T]/N)}$$

$$+ \frac{(1-\rho)\,(C\alpha_1/N)\exp(-\alpha_4 - \zeta E[T]/N)}{1 - (1-\rho)\,(\frac{N-1}{N})\exp(-\alpha_4 - \zeta E[T]/N)}$$

The total expected time to execute an M + 1 step sequence of instruction fetch and execution can now be derived. Recalling that only M' + 1 of these actions require memory access, the probability of j delays in the (M' - 1) - cycle sequence is

$$\Pr[j] = \binom{M'+1}{j} (\Pr[D])^j (1 - \Pr[D])^{M'+1-j} \quad .$$

The expected time to execute the M' + 1 cycle sequence when j delays occur is

$$E[S \mid j] = (M' + 1)\, E[T] + j\, E[W \mid D] \quad .$$

Hence the expected time to execute the sequence is

$$E[S] = \sum_{j=0}^{M'+1} Pr[j]\, E[S \mid j]$$

$$= \sum_{j=0}^{M'+1} \binom{M'+1}{M} Pr[D]^{j} (1 - PR[D])^{M'+1-j} ((M'+1)\, E[T] + j\, E[W \mid D])$$

$$= (M'+1)\, E[T] + E[W \mid D] \sum_{j=0}^{M'+1} j \binom{M'+1}{j} Pr[D]^{j} (1 - Pr[D])^{M'+1-j}$$

$$= (M'+1)\, E[T]$$
$$+ E[W \mid D]\, (M'+1)\, Pr[D] \sum_{j=0}^{M'+1} \binom{M'}{j-1} Pr[D]^{j-1} (1 - Pr[D])^{M'-(j-1)}$$

$$= (M'+1)\, E[T] + E[W \mid D]\, Pr[D] \sum_{j=0}^{M'} \binom{M'}{j} Pr[D]^{j} (1 - Pr[D])^{M'-j}$$

$$= (M'+1)\, (E[T] + E[W \mid D]\, Pr[D]) \quad .$$

This is again an intuitive result--the expected delay should simply be the expected delay per cycle times the number of cycles.

Expanding this result and substituting,

$$E[S] = (M'+1)\Bigg(E[T] + (1 - \Pr[A])\, E[W_c] + \sum_{k=1}^{\infty} \Pr[B_k]\, \Big(\frac{\rho C}{1-\rho} + E[T_p{}^k]\Big)\Bigg)$$

$$= (M'+1)\Bigg(E[T] + \frac{\rho C}{2(1-\rho)^2} + \frac{(\rho C\alpha_1/N)\exp(-\alpha_2 - \zeta E[T]/N)}{1 - \rho\left(\frac{N-1}{N}\right)\exp(-\alpha_2 - \zeta E[T]/N)} + \frac{(1-\rho)(C\alpha_1/N)\exp(-\alpha_4 - \zeta E[T]/N)}{1-(1-\rho)\left(\frac{N-1}{N}\right)\exp(-\alpha_4 - \zeta E[T]/N)}\Bigg).$$

In the case where input-output activity is absent (i.e., $\zeta = \rho = 0$), the above expression reduces to

$$E[S] = (M'+1)\left(E[T] + \frac{(C\alpha_1/N)\exp(-\alpha_4)}{1 - \left(\frac{N-1}{N}\right)\exp(-\alpha_4)}\right).$$

Barring a transfer of control during the sequence of instructions, the average execution time for M instructions is then $E[t] = E[S]/M$.

EXAMPLE

The ILLIAC IV Processing Unit [1, 2] will be used to demonstrate results from the foregoing analysis. Table 1 shows a breakdown of instruction types used in several ILLIAC IV codes (timing information in this and subsequent work is given in clock periods). Table 2 shows these instructions partitioned into the previously defined sets G' and G".

ILLIAC IV fetches 16 instructions simultaneously. It is assumed that half of these are Advast instructions which do not enter into the calculations. Hence M = 8 for this analysis and $Pr [g_0] = 11\%$. The probabilities of the other instructions have been reduced by one ninth to include the fetch cycle. Since memory reference and non-memory reference instructions are evenly divided, M' = 4.

It is assumed that the Final Instruction Queue (Finq) is kept sufficently full that the only delays are due to PE Memory conflicts. An excess of Advast instructions or such operations as LOAD and BIN are excluded in this analysis; their inclusion can only further delay the PE. The time to execute any pseudo-instruction, h_i, is given by

$$T_i = r_i + 5.5 \sum_{k=0}^{\infty} (0.57)\, k\, (0.43)^k$$

$$= r_i + 4.1 \quad .$$

The set, H, of memory-oriented pseudo-instructions, along with their probabilities of occurrence and average execution times is shown in Table 3. Assuming perfect memory overlap (i.e., memory access delays the PE only because of closely following sequential memory requests, rather than a lack of overlap access paths), an exact calculation of $Pr [B_1]$ (with $\zeta = \rho = 0$) and $E [T_p^1]$ is given in Table 4. From these, using C = 7 clocks, $\alpha_1 = 1$, $\exp(-\alpha_2) = 0.55$, $\exp(-\alpha_3) = 0.78$, and $\exp(-\alpha_4) = 0.43$ are calculated.

The expected duration of one block of instructions (again under the assumption of no input-output activity) is

$$E [S] = (4 + 1)(7.2 + 3.0)$$

$$= 51 \text{ clocks} \quad .$$

PE Operation	Probability	Number of Clocks Assuming Perfect Overlap
Add/Subtract		
from memory	10%	7
from registers	6%	7
Multiply		
from memory	9%	9
from registers	5%	9
Divide	2%	56
Load	18%	1
Register to register	18%	1
Route	10%	4
Mode Register Comparison and Bit Operations	9%	1
Store	13%	1

Table 1. ILLIAC IV PE Instruction Usage

Memory Instructions (G')			Non-Memory Instructions (G")		
Probability		Number of Clocks	Probability		Number of Clocks
g_0	11%	1	g_5	4%	7
g_1	10%	7	g_6	4%	9
g_2	8%	9	g_7	2%	56
g_3	16%	1	g_8	16%	1
g_4	12%	1	g_9	9%	4
			g_{10}	8%	1
Totals	57%			43%	

Table 2. Instruction Sets

Pseudo-instruction	Probability (%)	Number of Clocks
h_0	19	5
h_1	18	11
h_2	14	13
h_3	28	5
h_4	21	5

Mean execution time $E[T] = 7.2$ clocks

Table 3. Memory-oriented Pseudo-instruction Set

B_1 Events			Probability	Time to Complete
g_x		G'	39.0	6
g_x	g_9	G'	3.5	2
g_x	g_y	G'	9.4	5
g_x	g_y^2	G'	2.2	4
g_x	g_y^3	G'	.5	3
	Pr $[B_1]$	=	54.6	

Note: $g_x = (g_0 \cup g_3 \cup g_4)$, $g_y = (g_8 \cup g_{10})$

Mean wait if delayed = $E[T_p^1]$ = 5.5 clocks

Mean average delay = $E[T_p^1]$ $Pr[B_1]$ = 3.0 clocks

Table 4. Probability of Processor Self-interference and Expected Delay

This corresponds to an average instruction execution rate (over the set G) of

$$E[t] = 6.4 \text{ clocks per instruction}$$

as opposed to the ideal execution time

$$E[r] = 4.5 \text{ clocks per instruction} \quad .$$

Thus it is seen that the ILLIAC IV Processing Element efficiency is degraded by almost 43% due to the memory interference. Note that about one quarter of this is due to instruction fetching. Small kernels of code which can be entirely contained in the instruction look-ahead store are clearly advantageous.

Under the assumption that the exponential approximations used in the analysis are valid for the PE instruction distribution, the expected instruction block execution time for two independent memory banks per PE becomes

$$E[S'] = (4 + 1)\,(7.2 + 1.5/(1.0 - 0.21))$$
$$= 45 \text{ clocks} \quad .$$

The average instruction execution rate would then be

$$E[t'] = 5.6 \text{ clocks per instruction,}$$

with memory conflicts causing a 24% reduction in processor efficiency.

REFERENCES

[1] Barnes, G. H., et al., "The ILLIAC IV Computer," IEEE Transactions on Computers, Vol. C-17, pp. 746-757, August 1968.

[2] Davis, R. L., "The ILLIAC IV Processing Element," IEEE Transactions on Computers, Vol. C-18, pp. 800-816, Semptember 1969.

[3] Saaty, T. L., Elements of Queueing Theory, New York: McGraw Hill, 1961.

[4] Shemer, J. E. and Gupta, S. C., "A Simplified Analysis of Processor 'Look-Ahead' and Simultaneous Operation of a Multi-Module Main Memory," IEEE Transactions on Computers, Vol. C-18, pp. 64-71, January 1969.

[illegible]

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)		
ORIGINATING ACTIVITY (Corporate author) Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
	2b. GROUP	
REPORT TITLE Analysis of Simultaneous Operations and Memory Conflict In A Multimemory Computer System		
DESCRIPTIVE NOTES (Type of report and inclusive dates) Research Report		
AUTHOR(S) (First name, middle initial, last name) Luther C. Abel		
REPORT DATE March 15, 1971	7a. TOTAL NO. OF PAGES 30	7b. NO. OF REFS 4
. CONTRACT OR GRANT NO. USAF 30(602)4144	9a. ORIGINATOR'S REPORT NUMBER(S) CAC Document No. 5	
. PROJECT NO. ARPA Order 788	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
. DISTRIBUTION STATEMENT Copies may be requested from the address given in (1) above.		
. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Rome Air Development Center Griffiss Air Force Base Rome, New York 13440	

. ABSTRACT

This paper is an extension of the work of Shemer and Gupta on the ffect of multiple independent memory modules on processor efficiency and hroughput. Their conclusions, although valid, were unnecessarily restricted. he results derived herein expand them to include instructions of differing xecution times, instructions which do not require memory access, single as well s multiple instruction fetch and single as well as multiple word data fetches, lus the effects of a non-Poisson high-rate input-output device such as a wapping disk. An example of the application of the results to the ILLIAC IV omputer system is given.

D FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	
Computer Systems (General Purpose Computers)						
Control Units						
Storage Units						
Multiprogramming, Multiprocessing						

UNCLASSIFIED
Security Classification