Studies in the Organization of Computer Systems

Final Report

Michael J. Flynn

Research Program in Computer Systems Architecture

The Johns Hopkins University

Baltimore, Maryland

Report #34

June 15, 1970 to December 31, 1974

MASTER

PREPARED FOR THE U. S. ATOMIC ENERGY COMMISSION UNDER CONTRACT NO. AT(11-1)-3288

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Abstract

The project has studied theoretical and practical methods for the categorization, analysis and synthesis of computing systems. In particular, parallel processors were studied as to communications and processing efficiency; including an evaluation of stepping and splitting efforts. Later, the more general problems of interaction between concurrent processes were analyzed and a significant new computationally complete control mechanism was developed.

Recent project emphasis has focused on the development of an emulation laboratory for the analysis of computer architectures and certain language interactions. The project terminates at Johns Hopkins at the end of 1974.

Notice

"This report was prepared as an account of Government-sponsored work. Neither the United States, nor the Atomic Energy Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method, or process disclosed in this report.

As used in the above, 'person acting on behalf of the Commission' includes any employee or contractor of the Commission, or employee of such contractor, to the extent each employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor."

AEC FINAL REPORT AND PROGRESS SUMMARY
STUDIES IN COMPUTER ORGANIZATION

Introduction

The "Studies in Computer Organization" project at the Johns Hopkins University which was funded by the AEC will be a little more than four and a half years old when it terminates at the end of this year. A renewal of the contract is not being requested as the principal investigator will no longer be associated with the University as of the beginning of 1975. This report then is divided into three parts: the first part is a short and general review of the research and accomplishments that the project has achieved over the four and a half year period. The second part is a more detailed review of the research performed during contract year 1974. This is followed by a summary review of previously reported work and a list of supported research reports and publications.

A Four Year Review

By the end of 1974, the research project will have provided support for five students who have received their Ph.D., two Post Doctoral Fellows, and ancillary support for two associated faculty members, in addition to providing a significant amount of support for the principal investigator. Also, by the end of 1974 the project will have supported the publication of about 36 technical reports and approximately the same number of external publications.

The project has allowed the establishment of a small center for computer architecture studies; with emphasis on the following areas:

1. The theory of computing; the relationship between concepts in information theory such as entropy and computational complexity has been studied [Report #3,4,5].

2. General studies in computer organization and parallel processors.

In this area we have looked at the following specific problems:

- a) General categorization of processors and problems associated with each processor class [Report #12,14,16].
- b) Shared resource multiprocessor organizations [Report #2,10].
- c) Communications problems and certain kinds of parallel processors (master/slave organizations) [Report 19].

3. Theoretic analysis of concurrent/parallel processes. In this area we study the interaction between programs and processors: by modeling stochastic interaction and by developing and analyzing synchronization and control mechanisms for concurrent processes [Report #20,21,23, 25,26,32].

4. Emulation and microprogramming. A series of "soft" computer architectures were developed based on the advent of fast read-write control store; these were simulated and studied. A detailed machine architecture was specified and has been built to form the basis of our emulation laboratory. This laboratory is described later in this report and forms a unique facility for analyzing computer structures and the development of new architectures [Report #1,8,9,11,28,29,30,31].

5. Analysis of computer architectures. In the meantime through the use of trace tapes and simulation we have conducted a fairly extensive comparative review of classical computer architectures - the IBM 7090 and the IBM System 360. In this analysis we assess the architectural overhead; i.e., the incidence of instructions inserted in, or executed by, the programs which are principally due to the architecture of the processor as distinct from the program function [Flynn, IFIP CONGRESS].

The 1974 Annual Report

The latter three areas of research forms the basis for most of the project activities in 1974. In particular, significant progress has been made in understanding synchronization and control mechanisms in concurrent/parallel processes. Also, we will have completed (by the end of the year) our long awaited emulation laboratory which will provide us with a significant resource for future research in computer architecture. Some of this research - most notably in the area of directly executable languages and in the area of architectural studies has already begun - however, we look forward to the availability of the laboratory for expiditing future work.

1. Analysis of Concurrent Parallel Processes.

a) Control Mechanisms for Specifying Concurrency. During the above period, the investigation of Petri nets was continued by Mr. Agerwala. His most significant result is that an extension of ordinary Petri nets is complete: i.e., can represent all possible types of interaction between a group of processes and thus provides a correct formal counterpart to the vague notion of "coordination". Three models have been studied, each arising out of different motivations, and each apparently complete. The Turing machine model appeared to be complete because of the close relationship between the notions of "algorithm" and "coordination". A separate study performed by Mr. Agerwala earlier indicated that the extended Petri net model could be complete. Starting from the intuitive description of a coordination system, the SS (Synchronizing System) model was defined. This model also provided a generalization of all existing synchronizing primitives. Its generality indicated that it might also be complete. The equivalence of these three, independently motivated, apparently complete models was then established, thus indicating that each of the three models is in fact complete. Mr. Agerwala

has thus tried to formalize the intuitive notion of a coordination system. [Report #32].

Of the three models mentioned above, it is the extended Petri net model that is particularly interesting, as it provides a convenient tool to study concurrent systems. Coordinations can easily be represented and the nets are easy to understand.

Having established the completeness of the extended Petri net model, an analysis of controlling agents in all existing models of parallel computation was carried out. There were two motivations for this: most of the existing models have been proposed and studied independently and a thorough evaluation and comparison of the different models has not been carried out. Secondly, Mr. Agerwala wanted to show that most of the models had controlling agents (explicit or implicit) that were incomplete. The analysis was carried out systematically by establishing a hierarchy of Petri net models based on their capability for representing different kinds of interaction. Each of these models was essentially an ordinary Petri net model with a few modifications. Each existing model of parallel computations was examined, the underlying controlling agent was extracted and classified with respect to its position in the hierarchy (in most cases, this classification was direct because of the similarity of the controlling agent and one of the Petri net models). This study will shortly be published as a research report.

b) Complexity of Control Structures. J. Davison has been investigating sequential control structures from the point of view of feedback loop complexity.

It is assumed that there is available a given set of one input, one output functional modules, and binary test modules, capable of implementing some desired function. The control structure controls the sequencing of these modules by specifying the order in which they are

activated. In specifying these control sequences, any number of the given binary tests and functional modules may be used. These basic units may be combined into larger modules with the following restrictions: each module has precisely one input, at most K output connections, and at most R internal feedback connections. Any number of these larger modules may also be used in construction of other modules with the same restrictions on each module.

The question under investigation was: How do these restrictions effect the class of functions which can be implemented? He has found that it is possible to trade feedback connections for output connections, and that for a given maximum number of output connections, there are functions which require at least a specific number of feedback connections. In addition, modules with two output connections are considerably more powerful than those with only one. In fact, for any given number of feedback connections, there are functions which cannot be implemented by one-output modules, but which can be implemented by two-output modules, even if we restrict the two-output modules to a single feedback connection. This result appears to be extendable to modules with more output connections.

This work is being done in terms of software and structured programming, but the results are not restricted to this area.

2. Emulation Laboratory.

During the past year we have continued the development of our emulation research laboratory. This relatively unique facility is intended to serve as a research and educational tool in the area of machine architecture and system design. Specifically, this laboratory will support research directed towards the following goals:

- i. experimentation with a broad range of machines at low cost,
- ii. analysis of instruction streams of existing processors to determine areas of strength and weakness, and
- iii. implementation of "soft" machine architectures tailored to artifacts of high level, user oriented languages.

The central unit in the laboratory system is a microprogrammable host machine developed by C. Neuhauser [Report #28]. The control store of this machine is dynamic, that is writable in a time comparable to its read cycle, thus allowing it to be used as the primary storage resource in the system. Units in the system communicate with each other and with the host machine via a flexible bus system which allows units to access each other independently of the host machine. Control over the laboratory units is exercised by a processor independent of the host machine and oriented towards communication with the experimenter and facilities external to the laboratory.

Structurally the host machine is divided into three sub-machines, each receiving control information from the current microinstruction and operating independently, except when prevented by data dependent conflicts. One sub-machine provides operations designed to emulate the functional aspects of the target machine (e.g., arithmetic). Another machine controls the movement of information between registers, control store, and external memory. The third machine fetches the next microinstruction and provides conditional testing capability. Data paths within the machine are 32 bits wide, the control memory contains 4096 words, and eight general purpose registers are available.

The instruction set of the host machine has been arrived at by evaluating skeleton emulations of several target machines, including the IBM 360, and the DEC PDP-11. Field selection and manipulation instructions

have been shown to be especially important in the decoding process and in matching the resources of the host machine to those of the target machine. The parallel nature of the host machine resources has been used effectively and shown to be of value in the design of future machines. In particular, the evaluation has shown that the laboratory system is capable of emulating a PDP-11/20 at about one-third its instruction rate. With simple improvements to the main memory system it is expected to obtain real-time performance for this machine [Report #33].

While the host machine is being constructed by a consulting firm, we have been responsible for the design and construction of the interface between the control computer and laboratory bus system. This interface will allow the use of a control computer (a Datapoint 2200) to load host and target machine programs and to evaluate the results of emulation experiments. In addition a modification to model 38 TTY has been designed and constructed to allow the control computer to use the TTY's modem for communication with remote systems.

3. Analysis of Computer Architectures.

The use of computer architectures studied by examining the ratios gathered on a variety of executed programs. This was done for a technical environment for the IBM 7090 and compared with the IBM System 360. "Overhead" instructions include those instructions which merely rearrange a memory hierarchy and also those instructions which alter ordinary sequencing. The ratio of these overhead instructions to functional instructions are indicative of the use of the architecture. These ratios were developed for both 7090 and for System 360. The difficulty with projecting the fixed architectural arrangement to a variable user environment poses the key problem for computer designers. In partial response to these difficulties, more flexible architectures have been proposed which employ an interpretive level that matches new developments

in general storage structures for processors. This interpretive computing allows many aspects of the architecture to be altered to cater to a variety of environments. [Flynn, IFIP CONGRESS '74].

Highlights of Previously Reported Research [1970-73] - The following is a summary of some interesting research items which have already been reported in previous annual progress reports.

1. Shared Resource Computer Organizations. Simulations and analysis of skeleton processors sharing in space and time multiplex the execution and storage resources of a system. Some general constraints have been evaluated and the possibility of sub-commutation was investigated. In particular the problem of task assignment has been studied with a view toward minimizing the required monitor interaction in a multi-processor system. The proposed solution was the introduction of a dynamically reconfigurable FORK-JOIN instruction [Rpt.#4,14].

2. Studies in the Complexity of Physically Implemented Algorithms. One interesting hypothesis whose validity has been partially confirmed is that "on the average" the complexity or cost of implementation of a combinational circuit is proportional to the entropy of its truth table or map. "Entropy" in this sense is based on the probability of incurring a "1" in the truth table.

Other studies have investigated methods of relating the entropy concept to a metric function and hence to complexity [Report #5,6,7].

3. Our program was aided during 1970-72 by two Soviet computer scientists, Dr. V. I. Kartashev and his wife Dr. S. P. Kartashev, formerly of the Keiv Scientific Center (USSR) who served as Postdoctoral Fellows at Hopkins. Dr. V. I. Kartashev worked in the area of microprogramming and optimal computer organizations, while Dr. S. P. Kartashev worked on universal logic arrays and their relationship to the design of integrated circuits. She produced a number of interesting results on universal modules - logic circuits which realize other arbitrary functions by feeding back some permutation of the variables from the input of the original array. [Rpt. #15,17,19]

4. Graph theoretic analyses of program structures: Concurrent processes may be represented by bilogic graphs called procedure graphs and the determinism of the programs they represent is expressed by structural properties of these graphs. Determinacy in a program insures that a program is decidable. It is well known that "all" of the parallelism in a program cannot be detected - detection is an undecidable issue. Determinacy insures decidability. As part of the research reported on, Dr. R. Regis has shown that the determinacy property is decidable for acyclic graphs. Further, it is suggested that the standard notion of the determinacy is too restrictive. Consequently, two less stringent properties have been developed - partial and terminal determinacy and easily implementable decision procedures have been proposed. Further, recursively defined concurrent processes have been considered as well as their determinacy. [Report #21]

5. Stochastic analysis of parallel processing systems: In another study performed by R. Regis the operational characteristics of concurrently operating systems (multiprocessors) were analyzed and modeled as networks of multiserver queues. General techniques were developed to study the underlying stochastic nature of the models and to derive closed form expressions of their performance under rather general assumptions. Special attention was paid to the effect of scheduling strategies on performance measures, assuming arbitrarily distributed service time. A sensitivity analysis of the well known exponential assumption clearly showed its very peculiar nature and stressed the need for more general assumptions. This was a particularly significant development since the bulk of the technical literature on stochastic models of operating systems rely on this assumption. The fact that when parameters are not so distributed, significant departures from traditional systems design notions on batch processing vs. time-sharing systems, for example, can be demonstrated. [Report #20]

6. Dynamic detection of concurrency: Dr. T. S. Tjaden in another major study undertaken as part of the program investigated techniques for transforming a series of ordered (i.e. serial) tasks into their maximum parallel form. The transformations studied were restricted to those which were done during execution of the instructions and is distinct from a priori detection of the concurrency done by, for example, a compiler. A series of sophisticated techniques to accomplish these transformations were developed. These techniques used a multi-state ordering matrix, which detected instruction independence both intra-cycle and inter-cycle instructions. This matrix was extended to dynamically detect opportunities for reassigning the resources specified by certain instruction to increase the amount of potential concurrency.

Experimental results using sample program streams were developed. The results of these simulations were largely negative - showing strong relation to the incidence of the branch type instruction. The potential concurrency ~~developed~~ - even using these quite elaborate techniques - was much less than those reported for the a priori detection techniques. Thus, we feel that dynamic detection may no longer be a promising avenue for improved performance through concurrency in computer organizations. [Report #16]

7. Program structures and coordinations: As part of our continuing study of parallel processors, the parallel representation of programs and the coordination of concurrent cooperating task, Dr. Kosaraju completed a study [JHRR 25] on the limitations of the well-known Dijkstra's semaphores in specifying task synchronization. Patil of MIT had shown that the Dijkstra's primitives were not sufficiently powerful and proposed a generalization. Kosaraju shows that certain synchronization problems cannot be realized even with this generalization. Further he shows that an alternate approach to coordination, the Petri net, will be unable to handle some synchronization problems.

Despite this limitation, the Petri net provides a convenient method for representing the coordination between asynchronous processes. Mr. Agerwala [JHRR 26] studied two specific aspects of these nets: 1) capabilities and limitations and 2) "correctness". In the former study the notion of "~~interpretation~~" was introduced and two kinds of equivalence, strong and weak, of Petri nets were defined. Ordinary Petri nets were then modified in different ways and in each case it was determined whether the overall power of the nets, with respect to their capability of representing coordinations was increased or not. The final modified nets appear to be very powerful. In the second study Mr. Agerwala considered the problem of "correctness" of Petri nets and established the feasibility of using the methods of Computational Induction and Inductive Assertions to prove restricted statements about their nets. This study provides the possibility that one can mechanically translate a given system of asynchronous processes into a Petri net and then prove the correctness of the net. [Report # 25,26]

8. Parallel (SIMD) processing: In our investigation of parallel (SIMD) processing we have been examining three areas:

- 1) Analysis of communication structures,
- 2) Analysis of transformation units, and
- 3) Evaluation of the effects of instruction stream characteristics.

With respect to the analysis of SIMD communication structures, we have been concerned mainly with space-divided structures such as those used in ILLIAC IV. Of particular interest has been the class of structures called cyclic or star polygon networks. Our measure of communication structure effectiveness is the expected interstream distance (EID) which is a measure of the average time each communication oriented SIMD instruction will consume for a stated problem situation. We have found that star polygon structured networks have tolerably low EID when the particular SIMD problem requirement is 'regular', as in the matrix oriented calculations. However, when random communication requirements are imposed by a problem the EID increases greatly. For communication structures where the number of interstream paths is small with respect to N, the number of data streams, we have found that the EID will approach N.

SIMD transformation units (e.g. arithmetic/logic units) may be pipelined, vector structured or mixed. Mixed SIMD units, such as those in the CDC-STAR machine, are structured in both a pipeline and vector fashion. These three methods of structuring transformation units have been compared using as a figure of merit the utilization of available memory bandwidth. We have found that when memory bandwidth is fixed that pipelining reduces utilization

while vector structuring tends to increase it. The introduction of MIMD-like processing in pipelined units through the use of non-flushed pipelining techniques has been shown to greatly increase the utilization of memory bandwidth.

Instruction stream characteristics can strongly influence the average utilization of SIMD structured machines. We have evaluated two effects, stepping and splitting. Stepping occurs when the instruction stream contains instructions which use only one nth of the available bandwidth. Arguments have been made that for highly parallel machines utilization will be extremely low if even a small fraction of such instructions are present in the instruction stream. We have found that this argument does not apply directly to the SIMD organization when the average processed vector size is high, since the utilization of memory by vector instructions masks the effect of low usage instructions. We have also found that the introduction of low level MIMD type parallelism between the parallel structured resources and the sequentially oriented control unit can effectively offset the effects of low usage instructions.

Splitting refers to the reduction in an average vector length which occurs when data streams are separated into subsets by conditional testing. When different processing is required on each subset the SIMD structured machine must reduce the effective length of processed vectors. We have represented a restricted form of splitting by means of a Markov chain. The results of analysis indicate that the average processed vector length may be reduced to such an extent that SIMD performance with respect to a conventionally structured SISD machine will be logarithmically rather than linearly related to the number of potential data streams. [Report #10,12,19]

9. Microprogram minimization: This study was concerned with techniques for minimizing the representation of certain programs whose structure remains constant, microprogram "interpretive" routines belong to this class as was observed in the work of Schwartz, Grasselli and Montanari, and Glushkov. Glushkov's main aim was to construct an abstract model of an electronic computer based on an enlarged concept of an automaton. This abstract model enabled Glushkov to formalize a whole series of problems important from the point of view of optimization. One of these problems is the minimization of the control unit. Schwartz was concerned only with minimizing the bit dimension of a read only control memory of a microprogrammed computer. However, he was unable to provide an algorithm that guaranteed a minimal solution. Grasselli and Montanari reformulated the problem as a set covering problem of the prime implicant type. The ROM words $W_a, W_b, \dots, W_t$ were considered to be a set of subcommands: $W_a = \{S_{a_1}, \dots, S_{a_k}\}$, $W_b = \{S_{b_1}, \dots, S_{b_c}\}$, $\dots$, $W_t = \{S_{t_1}, \dots, S_{t_m}\}$. S_i and S_j are compatible if and only if $S_i \in W_h \Rightarrow S_j \notin W_h$, for all W_h . A compatibility class C_i is a class whose numbers are pairwise compatible. A minimal solution to the problem is any set of compatibility classes $\Delta = \{C_{i_1}, \dots, C_{i_n}\}$ such that: (1) every subcommand S_j is contained in at least one class in Δ , and (2) the quantity $B = \sum_{h=1}^n \log_2 (\#C_{i_h} + 1)$ is minimal. ($\#C_{i_h}$ = number of elements in C_{i_h}). Grasselli and Montanari achieve the minimal solution by solving a set covering problem exhaustively.

There is no known technique, other than exhaustive enumeration, to get a minimal solution of a set covering problem. Recognizing this fact, Mr. Agerwala presented a method that greatly facilitates the determination of a minimal solution. An implicit lattice is constructed where each node represents a solution. By exploiting certain properties of the particular problem at hand, only a small fraction of this lattice is examined and at each stage, an improved solution is obtained. The covering table method does not guarantee such an improved solution at each step. [Report #30]

10. Special purpose computer for emulation and simulation of computer

organizations: Associated with this project is the development of a small, very flexible "universal host" microprogrammable computer. This processor has three purposes:

- (1) To allow a user - by use of a read-write microstorage - to simulate hypothetical computer organizations and algorithms - e.g. parallel processors or algorithms for detection of simultaneously executable instructions, etc.
- (2) Limited ability for emulation of traditional organizations.
- (3) Provide a vehicle for research in directly executable languages (DEL).

The DEL serves as an intermediate level between a compiler from a higher level language and an interpreter into microinstructions.

Since an emulator is merely an interpreter of machine language, we are trying to expand the scope and power of the traditional "machine language" level. In particular, the use of read-write microstorage provides necessary speed and flexibility to make an expanded notion of DEL feasible.

At the outset of the project we had the following goals and limitations:

- (1) A cost of approximately \$5,000.
- (2) Emulation oriented instruction set.
- (3) Micro machine writable micro (control) memory,
- (4) Straight forward design to facilitate implementation, and
- (5) Compatibility with existing laboratory facilities.

Our preliminary efforts were concerned with determining the feasibility of building a reasonable machine within the cost objective above. Memory was clearly the largest cost factor (50-60%). After examining the various memory systems available it was decided that the machine characteristics should be as follows:

- Page 10
- (1) 32 bit micro instruction word,
 - (2) 32 bit main memory word, for compatibility,
 - (3) 1 K words (expandable) of dynamic MOS micro memory (180 nsec cycle time).

The basic logic type would be TTL because of cost, availability and ease of usage. MSI would be used where possible to reduce design and construction effort.

Basically, JHU-micro has a vertically organized instruction set with some facilities for low level micro parallelism. In designing the instruction set we have made use of our previous experience with micro machines designed and simulated at Johns Hopkins. Some important features of the JHU-micro machine are:

- (1) Flexible data shift and rotate facilities.
- (2) Several provisions for the introduction of immediate data into computations.
- (3) Special instructions to allow the isolation of specific fields within a data word.
- (4) Flexible memory manipulation capabilities.

Our experience with several short emulation programs written for JHU-micro indicates that the current design provides the facilities we desire for emulation.

In addition to the hardware design requirements we have specified a micro assembly language for JHU-micro. Where possible we have designed the micro assembly language so that it is compatible with our previous language designs. We hope that this will allow to make use for our previously written micro assembler with a minimum of modification. A report detailing the specifications of the JHU-micro machine, the micro assembly language and an example emulation is available [Report #28].

Principal Investigator

Michael J. Flynn, the principal investigator has charged 10% of his time to this contract with no charges occurring after May 1, 1974. At that time it was felt that all available financial resources had to be used to expedite the work of graduate research assistants, since termination of the project at The Johns Hopkins University became a possibility.

Disposal of Capital Equipment

Capital equipment items purchased under this contract are:

1. Model 38 teletypewriter
2. A Datapoint 2100 Terminal and
3. A specially designed computer for emulation research.

The Johns Hopkins University agreed to waive title to this equipment and it will be moved to Stanford University.

- JHCR #1: "A Simulator for a Universal Host Computer, It's Microassembler and a Partial 360 Emulator" by L. Yelowitz, J. Misera, S. Converse and R. Regis.

ABSTRACT: This study was undertaken as part of a seminar in Micro-programming given at The Johns Hopkins University during the fall of 1970.

The study proceeded in three phases - each described in a separate chapter of the report. The first phase consisted of defining a universal machine which in turn was simulated in FORTRAN on the IBM 7094. The universal machine was actually a concept of four storage resources and an execution resource which when parameterized would define a machine structure. No attempt was made to accomodate various data path widths, thus the 36 bits of the 7094 became a standard.

Chapter Two is a microassembler for the aforementioned universal machine and Chapter Three is an attempt at emulation of several System 360 instructions.

- JHCR #2: "Shared Internal Resources in a Multiprocessor" by M. J. Flynn. Presented at the IFIP CONGRESS 1971.

ABSTRACT: Some significant problems in a multiprocessor computer organization include, 1) development of an accelerating performance /cost relation with increasing numbers of processors, 2) variable task assignments to processors on an "as available" basis (and without continuous monitor interaction at task begin/end), 3) speed-up of a task execution when a strictly sequential (all others must follow) task is encountered.

Solutions to these problems are outlined by use of an organization using N "skeleton" computers, time-sharing the pipelined execution (arithmetic, etc.) resources. Since these processors are time-phased multiplexed with respect to the resources, a commutation pattern is developed which allows for subcommutation for fewer processors operating at correspondingly higher speed, when required. A "fork" to processors "as available" is introduced which requires a minimum of operating system intervention.

- JHCR #3: "Logical Network Cost and Entropy" by R. W. Cook and M.J. Flynn.

ABSTRACT: A measure of the minimum cost of a logical network is important in the evaluation of such networks. A hypothesis which states that the average minimum cost depends on both the number of input variables and the entropy ("entropy" in this sense is based on the probability of a "1" in the functions' truth table) of the function realized is investigated. Some experimental results are presented to support the hypothesis.

- JHCR #4: "Entropy Functions in a Metric Space" by R. W. Cook and M. J. Flynn.

ABSTRACT: A pseudo-metric space is defined for the set of all binary functions from n tuples to m tuples. The maximum conditional entropy of one function given the other is shown to be a pseudo-metric or distance function. By a redefinition of the set of functions, which is equivalent to considering a function and its' inverse as different representations of the same function, it is shown that the maximum conditional entropy is a metric. A generalized metric between any two finite functions is developed using the maximum conditional entropy.

JHCR #5: "Time and Space Measures of Finite Functions" by R. W. Cook and M. J. Flynn.

JHCR #6: "Towards Optimal Paginations for Structured Programs" by L. Yelowitz.

ABSTRACT: A type of programming known as "structured" is described and illustrated. A model for paginations is then presented for structured programs which bases its decisions for the locations of page breaks upon the syntactic structure of the program. This represents a departure from previous research efforts. A simple algorithm is given to solve the paginating model.

JHCR #7: "Methods for Realizing Sequential Machines on Identical Integrated Circuits" by S. P. Kartashev.

ABSTRACT: The development of integrated circuits has revised traditional views of switching theory. The problem of minimization of the number of memory or logical elements has yielded to the problem of optimal realization of a given sequential machine with integrated circuits.

In the report assignment techniques are developed for realization from identical integrated units.

In contrast to the majority of minimization methods the given methods do not require exploring many combinations, an important factor in their possible implementation.

In the sequel we introduce some definitions. Given a sequential machines $A(U, X, \delta)$, where U is the set of states, X is the set of input symbols, δ is the transition function. We presume that the outputs are unspecified. We are interested in the state assignment problem. Let U be the set of assigned states, X the set of assigned inputs. Each state is assumed to be coded into an n -bit number, and input into a p -bit number.

Let $F = (F_1, \dots, F_n)$ be the set of excitation functions which define the corresponding logical network. Each F_i ($i=1, \dots, n$) is the input to the i -th storage element and is a function of $p + n$ arguments. Functions F_i and F_j are said to be "similar" if a mapping U exists such that

$$F_i(y_1, \dots, y_p, x_1, \dots, x_n) = F_j(y_1, \dots, y_p, u(x'_1), \dots, u(x'_n))$$

for any vector $y = (y_1, \dots, y_p)$ from the input set X and for any vectors.

JHCR #8: "Microprogramming: Future Trends and Prospects" by M. J. Flynn.

JHCR #9: "Microprogramming: An Introduction and a Viewpoint", M.J.Flynn & R.F.Rosin.
ABSTRACT: The two broad aspects of microprogramming that are addressed in this issue are: microprogramming as a technological tool used to define and establish system control, and the application of microprogramming in the effective realization of computer systems.

As a technological tool, three areas seem especially significant in the control of a system: simultaneous internal resource management, the concept of residual control, and functional extensions of logic in storage.

The set of current application areas for microprogramming are many and varied, including the realization of computers and special devices, the emulation of earlier computers and operating systems, the support for user-oriented systems, system maintenance, and the enhancement of system performance.

JHCR #10: "Pipelining of Arithmetic Functions" by M. J. Flynn and T. Hallin.

ABSTRACT: Two addition and three multiplication algorithms were studied to see the effect of pipelining on system efficiency. A definition of efficiency was derived to compare the relative merits of various algorithms and implementations for addition and multiplication. This definition is basically defined as bandwidth/cost. Previous comparisons of adders and multipliers have generally been based on latency. In a pipeline environment, latency (or its inverse bandwidth) is not as important. Any bandwidth is possible up to the physical limitations on gate delay variations and pulse skew. The formal definition for efficiency is:

$$\text{Efficiency} = \frac{N}{D \cdot G}$$

where N is the number of bits in the operands, D is the delay (uniform) of each pipeline stage in units of gate delays and G is the total number of gates, including any used for latching. In cases where gate variations and pulse skewing are well defined pipelining using the Earle latch results in increased efficiency.

The most efficient adder is a maximally pipelined conditional-sum adder (3 stages with a delay of 4 gates per stage). Its efficiency is 6.30×10^{-3} . The most efficient multiplier is a maximally pipelined tree multiplier (8 stages with a delay of 4 gates per stage). Its efficiency is 3.48×10^{-4} .

JHCR #11: "Dynamic Microprogramming Processor Organization and Programming" by M.J. Flynn & A.B.Tucker. ABSTRACT: A dynamically microprogrammed processor is characterized by a small (4^k 64-bit word) read-write "micro" storage. The access time of this storage is similar to the cycle time of the machine (50-100 nsec). This microstorage is used to contain both data and subroutines. The (micro) instructions in such a processor differ from the conventional in that they perform only purely combinatorial operations; sequencing is under the control of the microinstruction. The presence of the read-write microstorage permits a more flexible assignment of resources than the read-only storage. In particular, the processor developed in this paper stresses the simultaneous operation (within the microinstruction) of the adder, shifter, masker, and testing facilities of the processor.

A microassembly language is developed and the overhead involved in sub-

routine linkages is analyzed. The efficiency of a flexible software linkage scheme is examined as to its overhead for various subroutine characteristics.

Finally, three examples of problem-oriented programming are considered and the resulting coding is compared against a System/360 assembly language version, with the technology normalized.

- JHCR #12: "Some Computer Organizations and Their Effectiveness" by M. J. Flynn.
ABSTRACT: A hierarchical model of computer organizations is developed, based on a tree model using request/service type resources as nodes. Two aspects of the model are distinguished: logical and physical.

General parallel or multiple stream organizations are examined as to type and effectiveness - especially as to intrinsic logical difficulties.

The overlapped simplex processor (SISD) is limited by data dependencies. Branching has a particularly degenerative effect.

The parallel processors (single instruction stream-multiple data stream - SIMD) are analyzed. In particular, a nesting type explanation is offered for Minsky's conjecture - the performance of a parallel processor increases as $\log M$ instead of M (the number of data stream/processors).

Multiprocessors (MIMD) are subjected to a saturation syndrome based on general communications lockout. Simplified queuing models indicate that saturation develops when the fraction of task time spent locked out (L/E) approaches $\frac{1}{n}$ (n , the number of processors). Resources sharing in multiprocessors can be used to avoid several other classic organizational problems.

- JHCR #13: "Annual Progress Report 1970-71 - Studies in Computer Organization".

- JHCR #14: "Modeling Generalized Parallel Computer Systems" by R. Regis.

- JHCR #15: "A Mathematical Model of Assignment For Realizing Modular Sequential Logic Networks" by S. P. Kartashev.

- JHCR #16: "Computer Resources - A Preliminary Study" by G. S. Tjaden.

- JHCR #17: "Theory of P-Multiple Regular Sequential Networks and its Application for Integrated Circuits I" by S. P. Kartashev.

ABSTRACT: This paper presents a new approach to the synthesis of sequential machines through integrated modules. The basic tools employed here are a concept of splitting shift register sequences to smaller ones and how this splitting affects next-state functions. The following problem has been solved: How does one find a proper state-assignment which enables the set $\mathcal{F} = (F_1, \dots, F_n)$ of excitation functions to be divided into p disjoint sub-sets $\mathcal{F}_1, \dots, \mathcal{F}_p$, where $|\mathcal{F}_1| = a_1, \dots, |\mathcal{F}_p| = a_p$

and $a_1 + a_2 + \dots + a_p = n$

such that the following substitution ψ on each of the subsets can be given: for any logical functions $F_i, F_j \in \mathcal{F}_S$, where $\mathcal{F}_S = a_S$ such a number $k \leq a_S$ exists that

$$\psi_k(F_i) = F_j, \text{ where } \psi_k = \psi(\psi_{k-1})$$

Thus if $k = a_S$, then $\psi_{a_S}(F_i) = F_i$. Hence any pair of functions $F_i, F_j \in \mathcal{F}_S$ ($S = 1, \dots, p$) is structure-invariant. That is a logical circuit realizing F_i can be utilized for realizing F_j by feeding some permutation ψ_k of variables from the input F_i . This capacity permits one to use a_S copies of one integrated mask in order to realize all functions from the set $\mathcal{F}_S$ ($s = 1, \dots, p$). In order to realize all functions from the set $\mathcal{F}$ it is necessary to have p integrated masks, each having a_i copies ($i = 1, \dots, p$).

JHCR #18: "Theory of P-Multiple Regular Sequential Networks and its Application to Integrated Circuits II" by S. P. Kartashev.

ABSTRACT: This paper is a further development of the results presented in part I. The questions under consideration include:
1. common technique for equivalent transform of a sequential machine M described by integrated format $P_1 = (a_1^1, \dots, a_p^1)$ to a form $\tilde{M}$ described by integrated format $\tilde{P}_1 = (\tilde{a}_1^1, \dots, \tilde{a}_q^1)$. Here P_1 and $\tilde{P}_1$ describe machines M and $\tilde{M}$ from integrated points of view: p, q are the numbers of different integrated masks, $a_i, \tilde{a}_j$ - numbers of copies for each integrated masks ($i = 1, \dots, p; j = 1, \dots, q$). Thus a common number of integrated modules necessary to realize machines M and $\tilde{M}$ respectively is:

$$n_p = a_1 + \dots + a_p; \tilde{n}_q = \tilde{a}_1 + \dots + \tilde{a}_q$$

M is called p -object, $\tilde{M}$ is called q -cover.

2. The following cases of this transform are considered:
 - (a) $p > q$, which is called iterative integrated reduction
 - (b) $p < q$, which is called iterative integrated expansion.
3. Numerical relationship for n_1 and n_q which constitute upper bounds on minimal amounts of integrated modules necessary to realize l-cover for a given l -object and q -cover for a given l -object respectively.

JHCR #19: "Communications in Parallel Processors" by Charles Neuhauser.

ABSTRACT: A method is given for analyzing inter-processor communications in parallel processor of the SIMD organization. Effectiveness of a particular communications structure is measured in terms of the expected time to perform an inter-data stream transmission of partial results. Results are applied to actual SIMD programs to determine the overall importance of communications in SIMD parallel processors.

JHCR #20: "Multi-Server Queueing Models of Multiprocessing Systems" by Renaud Regis.

ABSTRACT: In this paper, a general stochastic approach is proposed as a uniform basis for modeling multiprocessing systems and assessing their performance, under broad assumptions. Performance is evaluated in terms of global measures; the utilization factor of the servers, as a measure of external (user oriented) performance.

Of special interest is the influence of the queue discipline on these measures under the general assumptions: arbitrary arrival process and arbitrary service times distribution. We distinguish two classes of queue disciplines: preemptive and non-preemptive, and two types of arrival processes: dependent (on the state of the queue) and independent.

It is shown that, non-preemptive disciplines leave both measures invariant, for given but arbitrary (dependent or not) arrival process and arbitrary service distribution. For preemptive disciplines, the utilization factor is invariant only if the arrival process is independent, but the mean response time depends on the discipline, whatever the arrival process. It is also shown that the exponential distribution is the only distribution for which the queue discipline does not affect the performance of the system, thus underlining the very peculiar nature of this standard assumption and preventing us from safely generalizing the conclusions reached under this assumption.

- JHCR #21 "Structural Properties of Concurrent Processes" by Renaud Regis.
ABSTRACT: Characteristic of systems of concurrent processing activities operating in a common environment is their potential for non-deterministic (irreproducible) behavior. The notion of deterministic behavior is formalized in the context of a graph model of concurrent processes. In addition to complete determinacy, two less restrictive properties, partial and terminal determinacy, are introduced. Their decidability is constructively proved for acyclic graphs. Two extensions of the model are considered: recursively specified concurrent processes and mutually exclusive processes. With regard to the latter, it is suggested that they would be best studied in the context of partially interpreted models.
- JHCR #22 "Studies in the Organization of Computer Systems" by Michael J. Flynn.
- JHCR #23 "Representation of Concurrency with Ordering Matrices" by Garold S. Tjaden and Michael J. Flynn.
ABSTRACT: A formal technique for the representation of tasks such that the potential concurrency of the task is detectable, and hence exploitable, during the execution of the task is described. Instructions are represented as a pair of binary vectors, d and e, which completely describe the sources and sinks specified by the instruction. Tasks are represented as square matrices, M, called ordering matrices. The values of the elements of these matrices are used to dynamically indicate the necessary ordering of the execution of instructions.

It is shown how several different types of ordering matrices, each type having the capability of exhibiting different amounts of potential concurrency, can be calculated from the d and e vectors of the instructions of a task using "linear algebraic-like" operations. For example, inter-cycle independencies can be detected with a ternary ordering matrix. This matrix can be extended to dynamically detect opportunities for reassigning the resources specified by certain instructions to increase the amount of potential concurrency.

- JHCR #24 "Production Systems: A Formalism for Defining the Syntax and Translation of Practical Programming Languages" by Henry F. Ledgard.
ABSTRACT: This paper investigates the application of a formalism called production systems to define the syntax of a computer language and its translation into a target language. Several properties appear well-suited to this task:
- (a) The formalism can be used to specify exactly the syntax of a computer language, including context-sensitive requirements.
 - (b) The specification of the context-free portions of syntax and context-sensitive portions of syntax can be isolated.
 - (c) The same formalism can be used to specify the translation of a language into another.
- The following example applications of production systems are given:
- (a) A specification of the syntax of a significant subset of PL/1.
 - (b) A specification of the abstract syntax of the PL/1 subset.
 - (c) A specification of the translation of lambda-calculus expressions into normal form.
- JHCR #25 "Limitations of Dijkstra's Semaphore Primitives and Petri Nets" by S. Rao Kosaraju.
ABSTRACT: Recently various attempts have been made to study the limitations of Dijkstra's Semaphore Primitives for the synchronization problem of cooperating sequential processes. Patil proves that the semaphores with the P and V primitives are not sufficiently powerful. He suggests a generalization of the P primitive. We prove that certain synchronization problems cannot be realized with the above generalization. We also discuss the limitations of arrays of semaphores. We also show that even the general Petri Nets will not be able to handle some synchronization problems.
- JHCR #26 "Comments On Capabilities, Limitations and "Correctness" of Petri Nets" by Tilak Agerwala.
ABSTRACT: In this paper we examine the capabilities and limitations of Petri nets and investigate techniques for proving their correctness. We define different classes of nets where each is basically a Petri net with slight modifications and study the relationship between the various classes. One particular class appears to be quite powerful, with respect to its capability for representing coordinations. In the second part of the paper we establish the feasibility of using the methods of computational inductions and inductive assertions to prove restricted statements about Petri nets.
- JHCR #27 "Studies in the Organization of Computer Systems" by Michael J. Flynn (Progress Report)
- JHCR #28 "An Emulation Oriented, Dynamic Microprogrammable Processor" by Charles Neuhauser.
ABSTRACT: The design of a low cost, emulation oriented, dynamic microprogrammable processor is proposed. This machine is being built at Johns Hopkins University for use both as a research tool and an instructional aid. Features of this machine include: field isolation instructions, flexible access to microprogram writable control memory and convenient use of immediate data.

Reports #26 and #30 are available in revised form.

- JHCR #29: "Ideal" Directly Executed Languages: An Analytical Argument for Emulation by Lee W. Hoevel.
ABSTRACT: Several methods of evaluating user programs are analyzed with respect to space and time requirements. The concept of an "ideal" directly executed language is introduced, and it is argued that the "ideal: directly executed language for a contemporary computing system will not be either its source language or the language accepted by its base machine.
- JHCR #30: "A Survey of Techniques to Reduce/Minimize the Control Part/ROM of a Microprogrammed Digital Computer" by Tilak Agerwala.
ABSTRACT: In this report a survey of the research to date in micro-program minimization is presented. We discuss the works of Glushkov, Casaglia, et. al., Flynn and Rosin, Mischenko, Schwartz, Grasselli and Montanari and Das, et. al. The techniques are classified into three broad categories: the Glushkov Approach, the Ad Hoc or Engineering Approach and the Schqartz Approach. The author's views are summarized in the conclusion but the survey presents more than sufficient detail for the reader to draw his own.
- JHCR #28.1: "An Emulation Oriented, Dynamic Microprogrammable Processor, Version II" by Charles Neuhauser.
ABSTRACT: This report is an update of an earlier report and specifies the design of a low cost, dynamic microprogrammable processor intended primarily as a tool for research and instruction. Changes to the earlier report include:
1. Specification of extended arithmetic operations (e.g. multiply)
 2. Changes to method of accessing memory
 3. Specification of additional A-machine instructions
 4. Simplification of I/O structure
 5. Specification of host machine bus structure
 6. ISP-like description of the three internal sub-machines.
- JHCR #31: "The JHU Universal HostMachine II - Part I: The Machine" by J. W. Davison.
ABSTRACT: By "Universal Host Machine" is meant a machine on which it would be possible to "effectively" emulate (almost) any computer. This report describes an attempt to design such a machine based on a dynamically microprogrammable architecture. It is a two's complement, 32 bit, machine with a limited capability for parallel or overlapped use of internal resources.
- Part I of this report contains a description of the simulator and microassembler, as well as a description and partial evaluation of the micromachine. Part II (separately published) describes a partial emulation of the IBM 360 and further evaluates the micro-machine.
- JHCR #32: "A Complete Model for Representing the Coordination of Asynchronous Processes" by Tilak Agerwala.
ABSTRACT: The research presented in this report is directed towards formalizing the notion of a "coordination system". A model for representing coordinations is complete if it can represent all possible types of interaction between a group of processes, i.e. if it provides a formal counterpart to the notion of coordination. Three models are presented, the Turing machine model, the extended Petri

net model and the SS model, each arising out of different motivations and each apparently complete. The equivalence of these three models is established thus indicating that each one is in fact complete.

JHCR #33: "PDP-11 Emulation on the JHU Micromachine" by Stephen Bradley, Jr. and Charles Neuhauser.

ABSTRACT: A skeleton emulation of a DEC PDP-11 series machine on the JHU micromachine (see Report #28.1) is given and evaluated. The report contains the following sections:

1. Short description of the microassembly language
2. Skeleton emulation of the PDP-11
3. Description of emulator and technique used
4. Timing estimate and evaluation of micromachine performance.

This emulation indicates that the instruction execution rate of the emulated machine is about one-half to one-third that of a PDP-11/20. Architectural improvements are recommended which will allow the emulated machine to execute PDP-11/20 code in approximately real time.

JHCR #34: "Studies in the Organization of Computer Systems", Final Report by M. J. Flynn.

ABSTRACT: The project has studied theoretical and practical methods for the categorization, analysis and synthesis of computing systems. In particular, parallel processors were studied as to communications and processing efficiency; including an evaluation of stepping and splitting efforts. Later, the more general problems of interaction between concurrent processes were analyzed and a significant new computationally complete control mechanism was developed.

Recent project emphasis has focused on the development of an emulation laboratory for the analysis of computer architectures and certain language interactions. The project terminates at Johns Hopkins at the end of 1974.

JHCR #35: "An Analysis of Controlling Agents for Asynchronous Processes" by Tilak Agerwala.

ABSTRACT: The purpose of this report is to study different models of parallel computation. The capabilities and limitations of the controlling agents in these models are analysed formally. The approach is systematic rather than ad hoc. Nine different classes of nets are defined. Each class is essentially an ordinary Petri net with slight modifications. A hierarchy of these nets is established based on their capability to represent interactions among a group of processes. At one end is a class that is equivalent to finite state machines and at the other end, a class that can simulate arbitrary Turing machines. Almost every existing model is then analysed to determine where its controlling agent fits in this hierarchy. Apart from the fact that the report establishes the need for a unified theory and approach for the modeling of asynchronous systems some interesting results are:

JHCR #35: (continued)

1. The ordinary Petri net model, the complex bilogic, directed graph model, Petersons P-net model and Slutz's flow graph schemata model are equivalent and none of these is complete.
2. The extended Petri net model studied by us earlier, is one of the few complete models which is possibly of practical use and not introduced for theoretical definitions only.

1970 - 1974

T. Agerwala

"A Survey of Techniques to Reduce/Minimize the Control Part/ROM of a Microprogrammed Digital Computer", Seventh Annual Workshop on Microprogramming, October 1974.

"Comments on the Capabilities, Limitations and 'Correctness' of Petri Nets", First Annual Symposium on Computer Architecture, Gainesville, Florida, December, 1973.

M. J. Flynn

(with S. Szygenda), "Coding Techniques of Failure Recovery in a Distributive Modular Memory Organization", American Federation of Information Processing Societies (AFIPS) Conference Proceedings (SJCC), Vol 38, 1971, pp. 459-466.

(with S. Szygenda), "Failure Analysis of a Memory Organization for Utilization in a Self-Repair Memory System", IEEE Transactions on Reliability, May 1971, Vol R-20, No. 2, pp. 64-70.

"Microprogramming: Future Trends and Prospects", IEEE International Convention Record, 1971, pp. 318-319.

(with A. Tucker), "Dynamic Microprogramming: Processor Organization and Programming", Communications of the ACM, Vol. 14, No. 4, April 1971, pp. 240-250.

(with R. Rosin), "Microprogramming: An Introduction and a Viewpoint", IEEE Transactions on Computers, July 1971, Vol C-20, No 7., pp. 721-731 [a guest editor of this special issue on Microprogramming].

"Shared Internal Resources in a Multiprocessor", Information Processing 1971 N. Holland Pub. 1971, (presented at IFIPS Congress 1971, Ljubljana, Yugoslavia), pp. 565-569.

(with S. Szygenda), "A Perspective on Computer Reliability", 1972 Reliability and Maintainability Symposium, January 1972, San Francisco, Calif., Appears in Proceedings of 1972 Symposium of Reliability.

"Multiprocessors with Shared Resources", IEEE International Convention Record, March 1972, New York, pp. 356-357.

(with A. Podvin), "An Unconventional Computer Architecture: Shared Resource Multiprocessing", Computer, March/April 1972, Vol. 5, No. 2, pp. 20-29.

M. J. Flynn (continued)

"Towards More Efficient Computer Organizations", Presented at the Spring Joint Computer Conference, May 1972, Atlantic City, N. J., American Federation of Information Processing Societies (AFIPS) Conference Proceedings, Vol. 40, pp. 1211-1217.

(with T. Hallin) "Pipelining of Arithmetic Functions" presented at IEEE Computer Society's Workshop on Computer Arithmetic, May 1972, Col. Pk., Md., appears IEEE Transactions on Computers, Aug. 1972, Vol. C-21, No. 8, pp. 880-886.

(with T. Hallin), "Pipelining of Arithmetic Functions", (a version of the preceding paper) presented at the IEEE Computer Conference (COMPCON) Sept. 1972, San Francisco, California, appears in the COMPCON Proceedings.

"Some Computer Organizations and Their Effectiveness", IEEE Transactions on Computers, Sept. 1972, Vol. C-21, No. 9, pp. 948-960.

- above also appears in 1973 Best Computer Papers, I. L. Auerbach (Ed.) Auerbach Pub., Philadelphia, Pa., 1973, pp. 121-146. This paper was one of two selected as the best papers submitted to the IEEE Transactions on Computers during 1972.

(with S. Szygenda), "Self-Diagnosis and Self-Repair in Memory: An Integrated System Approach", IEEE Transactions on Reliability, April 1973, Vol R-22, No. 2.

"Computer Architecture at Johns Hopkins", Computer Architecture News (pub. ACM - SIGARCH) April 1972, Vol. 1, No. 2, pp. 21-33.

"Open Problems in Microprogramming", Proceedings (Supplement) of the Sixth Hawaii International Conference on Systems Sciences, Honolulu, January 1973.

"Open Problems in Microprogramming" (identical to the above paper) INFOTECH State of the Art Reports II.

"Microprogramming" and a number of shorter articles; Encyclopedia of Computer Science (A. Ralston, Ed.) Auerbach Pub., 1974.

(with G. Tjaden) "Representation of Concurrency with Ordering Matrices" IEEE Transactions on Computers, Vol. C-22, No. 8, August 1973, pp. 752-761.

(with R. Cook) "Logical Network Cost and Entropy" IEEE Transactions on Computers, September 1973, Vol. C-22, No. 9, pp. 823-826.

"Microprogramming Concepts", Advances in Programming (P. Freeman, Ed.) Science Research Associates (Pub.) 1974.

"Trends and Problems in Computer Organization", IFIPS Congress 74 (invited) Stockholm, Sweden, August 1974, to appear.

"Interpretation, Microprogramming and the Control of a Computer", Proceedings of the IEEE (invited paper) to appear.

"Microprogramming and the Control of a Computer" (similar to above paper), Chapter 10, Introduction to Computer Architecture, H. Stone (Ed.), Science Research Associates (Pub.) to appear

L. W. Hoevel

"Directly Executable Languages", 7th Microprogramming Workshop, Palo Alto, California, September 30, - October 2, 1974.

"Ideal" Directly Executed Languages: An Analytical Argument for Emulation, Johns Hopkins Computer Research Report #29, December 1973.

"Ideal" Directly Executed Languages: An Analytical Argument for Emulation, IEEE Transactions on Computers, August 1974, Vol C-23, No. 8, pp. 759-768.

S. P. Kartashev

"Integrated Circuit Realization of Sequential Machines (The Set of Algorithms)", Proceedings of the Sixth Annual Princeton Conference on Information Sciences and Systems, March 1972, pp. 245-432.

C. J. Neuhauser

(with M. Flynn), "Description of an Emulation Laboratory", Digest of Papers: COMPCON '74, Washington, D. C., September 1974.

"System Description of the JHU Emulation Laboratory", Proceedings of the 7th Annual Microprogramming Workshop, Palo Alto, California, October 1974.

R. Regis

"Performance Evaluation and Resource Allocation Policies in Multiprocessing", Abstract Preprints, IEEE Workshop on Parallel Processor, Architecture and Programming, Seattle, Washington, June 1972.

G. S. Tjaden

"Representation and Detection of Parallelism Using Ternary Matrices", Abstract Preprints, IEEE Workshop on Parallel Processor, Architecture and Programming, Seattle, Washington, June 1972.

L. W. Hoevel

"Directly Executable Languages", 7th Microprogramming Workshop, Palo Alto, California, September 30, - October 2, 1974.

"Ideal" Directly Executed Languages: An Analytical Argument for Emulation, Johns Hopkins Computer Research Report #29, December 1973.

"Ideal" Directly Executed Languages: An Analytical Argument for Emulation, IEEE Transactions on Computers, August 1974, Vol C-23, No. 8, pp. 759-768.

S. P. Kartashev

"Integrated Circuit Realization of Sequential Machines (The Set of Algorithms)", Proceedings of the Sixth Annual Princeton Conference on Information Sciences and Systems, March 1972, pp. 245-432.

C. J. Neuhauser

(with M. Flynn), "Description of an Emulation Laboratory", Digest of Papers: COMPCON '74, Washington, D. C., September 1974.

"System Description of the JHU Emulation Laboratory", Proceedings of the 7th Annual Microprogramming Workshop, Palo Alto, California, October 1974.

R. Regis

"Performance Evaluation and Resource Allocation Policies in Multiprocessing", Abstract Preprints, IEEE Workshop on Parallel Processor, Architecture and Programming, Seattle, Washington, June 1972.

G. S. Tjaden

"Representation and Detection of Parallelism Using Ternary Matrices", Abstract Preprints, IEEE Workshop on Parallel Processor, Architecture and Programming, Seattle, Washington, June 1972.