

SICMICRO

NEWSLETTER

SPECIAL INTEREST COMMITTEE ON MICROPROGRAMMING

VOL. 1 NO. 1

JUNE 1969

CONTENT

SICMICRO

**An explanation.....	3
A message from the chairman.....	5
A message from the editor.....	6
A review of the October 1968 Microprogramming Workshop held at Mitre.....	7
Trends.....	25
Calendar entries.....	27
Controversies.....	28

The SICMICRO Newsletter is an informal bi-monthly publication of the ACM Special Interest Committee on MICROprogramming. The scope of interest represented within SICMICRO include: System Architecture, Operating Systems, Hardware-Software Interface, Applications/Systems Engineering, Theory, and Logic Design.

The SICMICRO Chairman is S.S. Husson, IBM Systems Research Institute, 787 United Plaza, New York City, New York 10017, 212-983-7218.

Membership in SICMICRO is open to anyone interested in microprogramming.

Membership forms for SICMICRO are available from:

"SICMICRO"

ACM Headquarters

211 East 43 Street

New York, New York 10017

Changes of address or other matters pertaining to the SICMICRO mailing list should be directed to ACM Headquarters.

The SICMICRO Newsletter is edited by John R. Douglas, General Electric Company, MSD Systems Engineering B-124, 13430 N. Black Canyon Highway, Phoenix, Arizona 85029. Contributions may be sent directly to the editor. Letters in particular will be considered as submitted for publication unless they contain a request to the contrary. Sources of items published in the Newsletter will be clearly indicated, except for editorial items, which are contributed solely by the editor. The material published does not in any way reflect the opinions, philosophy, or policies of any of the employers of any contributor or any company or organization, including the editor unless clearly set forth as such. Rather the material solely represents the thoughts, opinions, and philosophy of the contributors or the editor. The publishers and SICMICRO officers are not responsible for not making any warranties or representations concerning the quality or accuracy of the material published.

AN EXPLANATION

It is not difficult to justify the existence of SICMICRO if you had the good fortune to be able to attend the Microprogrammin Workshop held in October in Bedford, Massachusetts last October.

The objectives and goals set by SICMICRO were stated concisely in a news release which we quote here:

"SICMICRO has the objectives of bringing together microprogramming practioners; providing them with a forum for the exchange of viewpoints, new ideas and problem solutions; and promoting the free exchange of information on all aspects of microprogramming."

To meet this set of objectives, SICMICRO has outlined a task for itself which entails a broad charter. Definitely planned for the coming year are the following programs:

1. A bi-monthly Newsletter which can provide a rapid vehicle for the publication of technical notes and comments.
2. Sponsorship of a tutorial session at an upcoming ACM Convention.

3. Sponsorship of the Second Annual Workshop on Microprogramming in October.
4. A Chairman of Technology has been chosen who will be responsive to the SICMICRO Membership who may wish to form working technical subcommittees to explore specific aspects of Microprogramming.

The initial areas visualized as generic areas of interest were as follows:

- a. Systems Architecture
- b. Logic Design
- c. Operating Systems, Hardware/Software Interface
- d. System Engineering/Applications
- e. Theory

If there is a question as to where your particular interest should fit - or if your particular interest deserves more specific partitioning - the Newsletter Editor would appreciate your comments and will pass all suggestions on to the Technology Chairman.

Harold W. Lawson, Jr.
Polytechnic Institute of Brooklyn

--the Editor

MEMO FROM SICMICRO CHAIRMAN

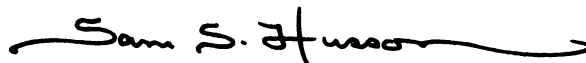
May 5, 1969

Dear Prospective Member:

Thanks to the many people who have supported this movement. We now have the opportunity to contribute and see the SICMICRO flourish and fulfill our objectives. Our main objective was to bring together microprogramming practitioners; providing them with a forum for the free exchange of information on all aspects of microprogramming.

Your support can come in many forms. It can come by your active support and involvement in the Second Annual Workshop on microprogramming, which will be held in Phoenix, Arizona on October 13 and 14; or by your contributions to this newsletter or by holding tutorial sessions at your local ACM or IEEE chapters, and finally by contributing articles and papers to technical journals and conferences.

Your committee, recognizing the diversity of interests and points-of-view has taken the challenge to characterize and define the different needs of the different interest groups. We recognize the logic designer to whom microprogramming is but an alternative to the traditional hardware control. We recognize the needs of the application man and the manufacturer's technical liaison to the customer who views microprogramming as another option to design a system which is architecturally more suited to the user's requirements. We recognize the software man, the operating system designer and programmer who by the proper utilization of this option can reduce the overhead implied in the general operating system. Finally, we recognize the academic and the theoretician who can lay the solid theoretical foundation to this field from its infancy, and who can use this systematic, pedagogical approach to the system architectural implementation, to teach a number of otherwise difficult concepts. In short, we do recognize the diversity of interests and opinions. It is possible that we may have missed other interested groups. If so, please let us know; but in any case, we do want to have all these viewpoints represented and interactive. And it can only do so by your active support and contributions.



Sam S. Husson
SICMICRO - Chairman

SSH:mfc

A MESSAGE FROM THE EDITOR

I have accepted the post of editor presupposing I will have something to edit, which puts the onus on you, good reader, to involve yourself to some point in this newsletter. If you have a subject area in which you would care to expound, or just try a halfbaked idea out for size, send a copy in, it doesn't have to be the polished(?) paper seen in a Spring or Fall Joint Conference. If you would rather not involve yourself to the point of doing an article, let us know what you would like to see expounded, and by all means read, discuss, and disagree with what you see here. That is exactly the purpose of a forum of this kind.

In early issues, we have the following materials already lined up for publication:

- an article on emulation
- a comparison of Spectra VS 360 (as seen by a microprogrammer)
- a bibliography of microprogramming references

This paper describes the Workshop on Microprogramming, sponsored by the ACM with the cooperation of MITRE which took place on October 7-8, 1968.

It is primarily a historical account and summary of the Workshop with some commentary from the author, Joseph E. Sullivan of MITRE Corporation. The Author wishes to acknowledge T.L. Connors and B.J. Huberman for their advice and assistance in the collection of the material for, and the preparation of the following account.

SECTION I

1.0 INTRODUCTION

The Workshop on Microprogramming, sponsored by the ACM with the cooperation of MITRE, took place at MITRE on October 7-8, 1968. The purposes of the workshop were to:

- Identify the work currently being done in the field
- Promote communication between microprogramming practitioners
- Identify promising new directions for investigation

Ninety-one individuals attended the Workshop: fifty-seven from industry, twenty-three from the academic world or university-associated laboratories and eleven from non-profits or government. Eighty-nine were from the United States, with one each from Italy and Great Britain.

The Workshop was conducted in four sessions. The first three were relatively formal, with scheduled speakers, and the fourth a less formal discussion session.

1.1 Current Work

Personnel from some seventy-three projects, more or less, were represented. These could be classified roughly as follows:

- a. General Study or Teaching of Microprogramming (17)
- b. The Design of Support or Automation Processes for Microprogramming (10)
- c. The Use of Microprogramming for:
 - Higher-Level Language Support (8)
 - Emulation or Extension of Instructions Sets (11)
 - Direct Applications or Process Control (13)
 - Machine Architecture (14)

1.2 Communication

Several of the speakers alluded to microprogramming as a "bridge" between hardware and software. When someone schooled in hardware talked about hardware, and when software experts discussed software, the conference indeed served as a bridge of communication between the two groups. However, whenever interdisciplinary aspects of microprogramming were considered, the two points of view tended to diverge. In these cases, the workshop served as a forum for debate and as a medium for mutual astonishment, if not enlightenment, between the hardware and software schools. Even the definition of microprogramming was not resolved because, of course, each tradition sees this new technology as an extension of, and in terms of, its "old" technology - and the terms differ. The subjects of who should do microprogramming, how he should do it, and what he should use it for, likewise touched off much controversy, especially in Session IV. It should be remarked that the present writer is of the software school.

1.3 New Areas of High Potential Payoff

Emerging from the workshop were two central themes that, though not really new, will clearly become more important as the state of the art advances.

One of these might be called "Tradeoff Technology". While several papers were delivered touching on the subject, and while anguished cries would be heard for "Guidelines" on the use of microprogramming, it was fairly clear that no practical work had been done towards establishing a set of rules for drawing near-optimum lines between hardware, firmware, and software portions of a system. Almost by definition, a significant payoff will reward anyone who can dent this problem.

The other subject that kept coming up was writable control stores. The topic generated much controversy (Session IV) but it is obvious that writable control stores will be produced and marketed as soon as the technology and market support them - and the market, it appears, is waiting. The proper use of this facility, so as to extract from it a maximum measure of the computer power and flexibility it seems to offer, is a virtually unexplored technology.

SECTION II

2.0 SESSION I

"THEORY OF MICROPROGRAMMING"

Chairman: Professor H. Gray
University of Pennsylvania

Microprogramming - interpreted as implementing control logic, primarily by read-only storage - cuts across the specialties of electronic module design, logic design, mechanical languages, programming and system architecture. Microprogramming is, therefore, a promising means for designing integrated hardware-software systems. In this session we will discuss topics fundamental to microprogramming and these related specialties.

-- from the Workshop Bulletin

M. V. Wilkes

The Growth of Interest in Microprogramming

Appropriately enough, the first session of the workshop was opened by the man generally credited with introducing the term microprogramming in the sense of "implementing control logic by a control store - usually a read-only store".

Professor Wilkes attributed the origins of microprogramming to dissatisfaction with less systematic methods of system design and implementation. He traced the history of microprogramming through an early period of interest (early 1960s) and through successive periods of declining and (now) resurging interest. The intermediate decline he attributed to a general lack of successful application in the early days, due to (1) the lack of a fast and cheap ROS, and (2) the shift in emphasis away from machine level efficiency that accompanied a growing acceptance of higher-level languages. The present renewal of interest, he claimed, is due to advances in memory technology and the influence of the IBM 360 architecture, which demonstrates the feasibility of implementing similar instruction sets in small and large machines.

Professor Wilkes recalled his earlier opposition to modifiable control stores - an opposition based on an expected "chaos" that would result if every programmer had his own instruction set. He softened this position only to the extent that modification of the control store is reserved for special "system" programs or programmers using now well-understood protection techniques.

A. Tonik

Tradeoffs for When and How to Use Microprogramming

Mr. Tonik contrasted the "microbird's eye" view of a processing system as consisting of adders, shifters, etc., with the other view which sees it as an elaborate array of logic - gates, decoding networks, and the like. He listed the liabilities of the microprogramming approach as (1) a slower cycle due to the need for control memory access, and (2) an increase in hardware (including, presumably, the control memory itself). To substantiate the latter, he estimated that 900 modes in a small conventional machine are equivalent to 50,000 control memory diodes, and that 8,000 modes in a large conventional machine must be replaced by 300,000 bits of control memory to obtain an equivalent microprogrammed machine - plus, of course, the microprogram control circuits in both cases.

The standard list of advantages for microprogramming as a machine design tool was discussed: the ability to add or modify instructions, precheck a design by simulation, emulation, etc.

The architecture of the micromachine itself was also treated at length. One of the questions discussed was the best place for keeping the address of the next instruction - (1) in the instruction word itself or, (2) in a separate counter. The figures quoted were: a 15-35% larger word results if method (1) is followed; 10-25% more words (the unconditional branches) are used with method (2). Although the second alternative thus seems to be preferable in most circumstances, Mr. Tonik pointed out that one-instruction subroutines and a 5-10% improvement in running speed can be realized with the first method.

On the subject of writable control stores, Mr. Tonik pointed out the possibility of dynamic modification of the control, but said he was not in favor of it. (This, of course, is not quite the same thing as "user" modification of the control store in a non-dynamic fashion.)

Y. Chu

A Higher-Order Language for Describing Microprograms

Dr. Chu described a language suitable for the description of a micro-machine - its hardware registers and state transitions. A broadbrush description of the language's main features and syntax was given, and a sample "program" in the language was exhibited. A simulator (for testing micromachine architectures), which accepts the language as input, has been constructed.

Although it would not be possible to evaluate the language itself from the level of detail in the talk, it is clear that such a language is valuable not only as a tool for design and design testing, but also as a means of rigorously defining a machine to microprogrammers - that would be a significant advance over functional (English) machine descriptions and/or wiring diagrams - particularly if used to supplement them.

G. Y. Wang

Micro-program Memory Technology

Mr. Wang discussed current developments in high-speed memory technology. The read-write characteristics of the technologies discussed are summarized in the following chart, taken from the first slide of the presentation.

<u>Technology</u>		<u>Destructive Read-Out</u>	<u>Non-Destructive Read-Out</u>	
			ROM	Electrically Alterable
Magnetic Core	1 core/bit	X	X	
	2 core/bit	X		X
	multi-aper- ture (BIAX)			X
Magnetic Thin Film	plated wire	X	X	X
	flat film	X	X	X
	mated film			X
Solid-State LSI	MOS		X	X
	Bipolar		X	X
	Diode Array		X	
Linear Coupler	Inductive		X	
	Capacitive		X	
	Resistive		X	

User Microprogrammable Computers

Professor Ramamoorthy defined a "user microprogrammable computer" as one in which the user has some (restricted) access to the microprogrammable store. He distinguished two classes of users: (1) those who own the machine system and (2) those who solve their problems on the system. The first group he described as interested primarily in maximum utilization and also such specific problems as system expansion/changeover transition problems, emulation, user-user and user-system protection, automated surveillance, and the like. The second group he characterized as interested mainly in maximum convenience, the "naturalness" of the languages provided, the production performance, turn-around and other aspects of program production, and possibly, real-time dependency of the problem program.

The user of either class with the desire and fortitude to formulate his own solutions to his problems can make good use of microprogramming, particularly from a timing standpoint. However, it was pointed out that merely giving access to the microprogrammed store is not enough; the following requirements (or, at least, desirable characteristics) were also listed:

- 1) A writable control store
- 2) Reasonable user cost
- 3) Simple micro-commands
- 4) A language capable of describing micro-operations
- 5) Relocatability and reentrancy of user microprograms
- 6) Flexible addressing (e.g. vector, matrix, proximity)
- 7) Simplicity of the "visible" machine
- 8) Parallel surveillance and user-user, user-system protection

L. L. Rakoczi

Implications of Dynamically Changeable Microprogram Memory

Mr. Rakoczi gave a spirited presentation on what he called the "fourth generation" computer architecture: the "machine within a machine" - that is, a dynamically modifiable control store. Used in a more or less obvious way, such stores permit multiple emulators to be run on the same processor under a central control and with very little switch-over time. Mr. Rakoczi cited several successful emulation projects along these lines. More exciting was the "6000-E" machine, which actually permits something close to dynamic modification of the control store (with restrictions, of course). The assembler for this machine allows new instructions to be defined in terms of the microcommand sequence which realize them; these get loaded into the control store when the "native" level program is loaded into the main store.

The talk touched off a lively discussion on the subject of writable control stores; it became apparent for the first time that opinions on the matter differed widely and that the subject would occupy much of the workshop's attention.

SECTION III

3.0 SESSION II

"MICROPROGRAMMING PRACTICES AND TECHNIQUES"

Chairman: Dr. R. Merwin
DOD

This session will concentrate on the mechanics and techniques for using microprogramming to implement hardware projects. Main topics of discussion will be: Control Store Designs, Simulation of Microprogrammed Systems, Techniques for Writing Machine-Descriptions, and the writing of associated Microprograms.

Hardware and software tools required for these tasks and descriptions of specific approaches to microprogrammed computer systems will be discussed. Tradeoffs between control and operational hardware and a transformation technique for evaluating these, based on microprogramming techniques, will also be covered.

---- from the Workshop Bulletin

G. E. Hoernes and L. Hellerman

Experimental List-Type Micro-Code Assembler

Messrs. Hoernes and Hellerman discussed the motivation for, and characteristics of, a "list-type" assembler to supplant the "box-type" assemblers now in use at IBM. The intention is to utilize "conventional" techniques and to ease the transition from higher-level to micro-code for applications programmers with a need to perform optimization.

The assembler described is itself coded in PL/I. It has "equation"-format input and a number of output options to satisfy the needs of various users: the programmer who wants to see the logic flow, the engineer who wants to see what bits are set in the resulting control word, and the customer engineer who may want both.

The talk dealt mainly with problems peculiar to IBM's own environment. For example, terms such as "edge characters" - meaningless except in the context of IBM's "box" assemblers - were used. On the whole, though, the talk was fairly well received.

J. R. Vollbrecht

Microprogramming Design Aid System (MIDAS)

MIDAS, as described by Mr. Vollbrecht, appears to be a standard set of microprogramming utilities - an assembler, a simulator, a flow charter, a manufacturing data puncher, etc., - remarkable mainly in that they are organized under a coherent "control" program. Well-established software utility principles seem to have been applied with considerable success.

G. Hoff

Development of a Microprogramming System for the H4200

Mr. Hoff's presentation was a fascinating (to a programmer) view of microprogramming from a logic designer's standpoint. The machine described - the H4200 - was logic designed and microprogrammed by the same people, and the impression was that these were not, primarily, programmers. Consequently, the project could be considered a case study of the processor architecture that results under such circumstances.

The most significant design constraint was that the H4200 requires a high degree of parallelism in order to keep up with the memory while processing punctuation-delimited data fields. The micromachine command structure arrived at to satisfy this requirement might be described as a "45°" machine partaking of some "horizontal" (one-bit-per-gate) qualities and some "vertical" (bus or register-oriented) qualities. It had, for example, curious six-way branches which do not occur until after the instruction following the branch has already been executed.

E. Stabler

Microprogram - Micro-Operation Transformations

Professor Stabler described an automatic process for determining appropriate state reductions - in effect moving logic elements from the control unit to the operations unit of a computer. The method represents a theoretical approach to the important practical problem of balancing hardware-software tradeoffs.

It was not clear that the method had practical application in its present form, inasmuch as the sequential-state description of a typical process (say, a floating-add) in a real computer would be a formidable task, and in any case one cannot be sure that the reduced process is unique and independent of the original detailed process. Nevertheless, the study has potential practical use as well as theoretical interest.

G. S. Badger

The Use of Microprogramming in a Course in Computer
Operating Systems

It would seem that a first course in operating systems - on a "dirty" machine - would be hardly the place for the subject of microprogramming to come up. Mr. Badger's reasons for making the study of microprogramming to come up. Mr. Badger's reasons for making the study of microprogramming an integral part of such a course summarize the view of microprogramming as a medium of communication, especially pedagogical communication:

- 1) The breakdown of instructions into micro-instructions gives some idea of instruction commonality.
- 2) The principles of interpreter architecture are illustrated.
- 3) A non-hardware description of the machine is afforded by the microprogram.
- 4) An idea of what is time-consuming for the machine to do may be gained from putting together a microprogram.
- 5) The logical arbitrariness of the hardware-software "line" is illustrated, and the practical considerations that go into drawing this line are more fully appreciated after a microprogramming exercise.

H. C. Forsdick and Dr. R. Merwin

Microprogram Control Design and Simulation System

Dr. Merwin, speaking for Mr. Forsdick, described a system composed of a hardware description language, actual microcode language and simulation language, useful as a pedagogical tool and design experimentation aid.

SECTION IV

4.0 SESSION III

"APPLYING MICROPROGRAMMING TECHNIQUES"

Chairman: Mr. J. D. Babcock
Allen-Babcock

(represented by P. R. DesJardins, Allen-Babcock)

The advent of read-only-store computer design has suggested the ability to "extend" the use of such a machine beyond that of a general-purpose production tool. This session will concentrate on projects in which the use of micro-programmed machine resulted in an extended-machine (as opposed to a limited-machine) design.

The main discussion will present specific projects, including statistical reports on the application (thus, why was it microprogrammed in the first place), costs of the programming effort, and the required training and background needed for the implementors. Some emphasis will be directed toward anticipating future needs in implementing microprogramming application. This information will be based on the experience gained on the projects described.

--- from the Workshop Bulletin

P.R. DesJardins

The Use of Microprogramming to Enhance Machine Performance of a Time-Sharing Programming System

Mr. DesJardins described an extension of twenty-two instructions to the 360/50 manufacturer-supplied set. The instructions described were in support of an interactive time-sharing system (RUSH). Those singles out as having the most payoff were variants of a chained list search. Other 'list' instructions, "reentrant utility" instructions, and genuine floating decimal instructions were also discussed. (The microcode is available from IBM as RPQ's.)

D. Boyle

Microprogramming of Data Logging and Data Reducing Equipment

Mr. Boyle described a simple processor capable of driving and sequencing the steps of a physical experiment. The system could be characterized as a sort of "immediate" process control, or a computer with radars, A-D converters, and the like as components but largely lacking in the usual things such as memory or arithmetic units. Mr. Boyle himself said the system was simple and did not claim that a sophisticated or advanced application of microprogramming was involved. Some discussion was triggered by a remark from I. Flores to the effect that the system was too simple to be interesting and not even microprogramming - a view not generally supported by those present.

C.L. Mathis

Emulating the 7094 on the 360/85

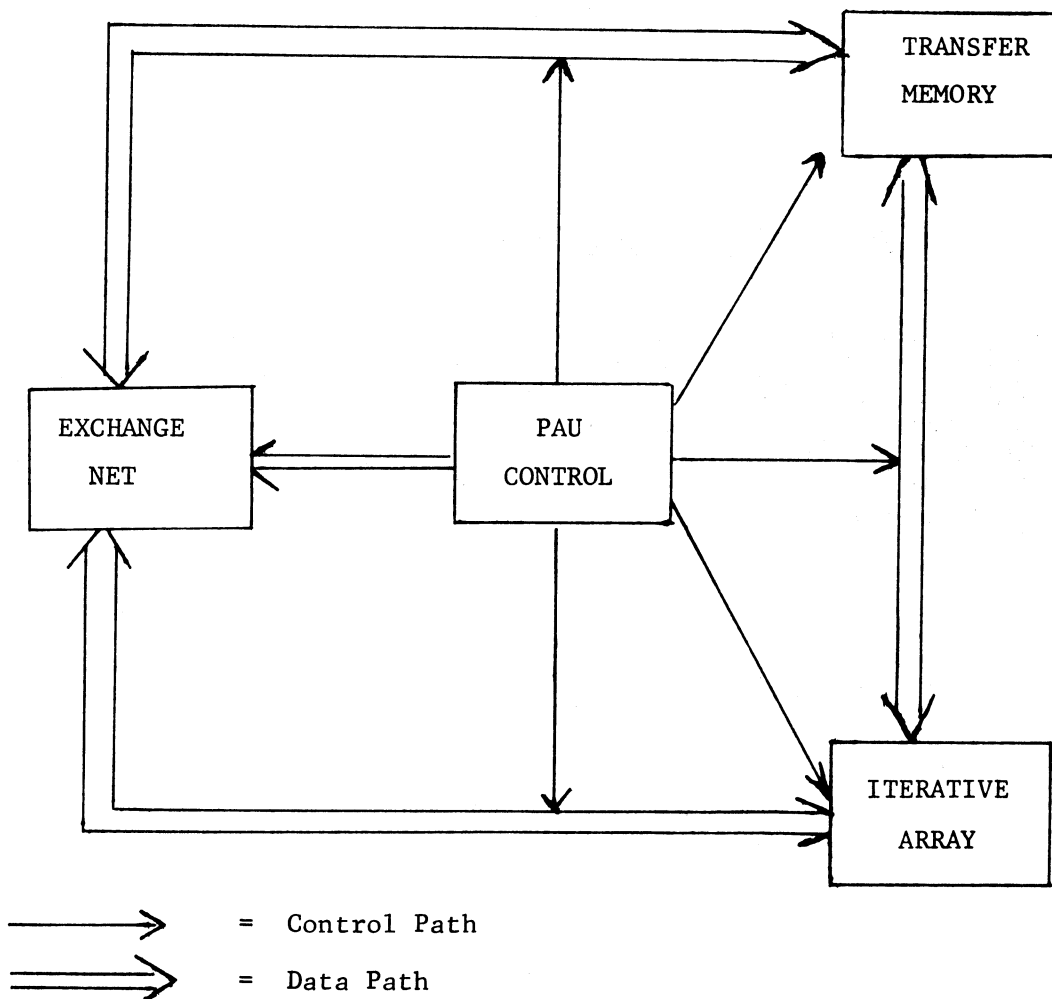
Mr. Mathis of IBM described various approaches to the specific problem of 7094 emulation on the 360/85, with a performance goal of doubling the 7094 speed. Two approaches were discussed, (1) a "subroutine interpretive mode" involving a slight extension of the 360/85 instruction set for decoding 7094 instruction and executing a few of the more frequently used ones, and (2) "microprogram interpretive mode" wherein sequences of 7094 instructions may be executed in microcode. Approach (2) required various ad hoc additions to the 360/85 hardware, for example, an inhibit of the instruction look-ahead.

R. T. Borovec

A Report on the Use of Microprogramming for the Illiac III Image Processor

The session opened with a presentation by R. T. Borovec on the Illiac III Image Processor. Developed for high-energy physics applications, it is now used also in biological and weather photo analysis. The central control point or "pattern articulation unit (PAU)" is microprogrammed; this unit was the focus for most of the talk.

Interesting features of the Image Processor are a transfer memory accessible either as 1024 48-bit words or 48 1024-bit words and a 32 x 32 bit "iterative array" which may be used as an associative memory. The organization may be depicted as follows:



B. Caruthers

Microprogramming to Support FORTRAN Functions

When an instruction set is implemented without "excess garbage" - from a FORTRAN point of view - the result is a faster running FORTRAN. This perhaps predictable consequence was documented by Mr. Caruthers in his discussion of such an instruction set. Speed was improved by the following factors:

- < 2 in general computation;
- 5 to 7 in subscript calculation;
- "hundreds" in computation of exponentials.

H. Lawson

Microprogrammed Higher Level Language Computers

Mr. Lawson opened with a set of definitions (with credits to Julien Green) that included: "Microprogramming is programming the interpreter". Mr. Lawson went on to present the case for microprogrammed truly higher-level languages. The reasons he advanced and the general properties he postulated for such a language seemed to imply something on the order of PL/I or even beyond. The benefits to be derived were stated to be as follows:

- (1) Simplicity - fewer levels of language to cope with;
- (2) Efficiency - "super" operations may be performed in high-speed control;
- (3) Compatibility - historically, more success has been realized with higher-level languages than with machine languages.

SESSION V

SESSION IV

"SELECTED SHORTS ON THE PAST AND FUTURE"

Chairman: Mr. A. Siegel
Decision Systems

This session will assess the extent and direction, if any, of the influence of microprogramming concepts upon the design and utilization role of user-prepared microprograms, the relationship between engineering and programming, the implications of writable control storage, and the feasibility of standardizing micro-processors and/or microprogramming languages.

Based upon their own experience, all participants will be encouraged to express their views on the benefits or dangers of the more widespread use of microprogramming. The discussion is expected to culminate in meaningful conclusions concerning the future of microprogramming and its role as a lasting influence, or a passing fad, in computer technology.

Since vigorous discussion is anticipated, participants who wish to ensure themselves of an opportunity to speak may do so by contacting the session chairman prior to the meeting.

- from the Workshop Bulletin

Proceedings

Session IV was by design an open-floor session with only short, informally scheduled and informally delivered talks from the rostrum.

Dan Zatyko of General Electric lucidly illustrated the problems that arise when microprogramming is defined in terms of language level - e.g. as "implementing control logic" - because, of course, any level can be thought of as interpreting and executing a "language" at a higher-level. He referred to the level below microprogramming as "picoprogramming" - a term which hasn't caught on, at least not yet.*

Julien Green characterized micromachines by such things as their sparsity of registers and the treatment of main memory as an I/O device.

Still other definitions of microprogramming were advanced, some in terms of hardware ("regular" vs. "irregular" memories), and others in terms of characteristics such as parallelism. This writer offered the view that "micro" is, after all, a quantitative prefix and that cannot be drawn between "bit" and "little". This failed to settle the question, but at least the discussion moved on to other questions soon thereafter.

Bob Rosin of SUNY exhibited the architecture of a hypothetical computer he uses for pedagogical purposes and for which a simulator has been built.

C. Billings of Honeywell presented the case for a higher-level language to be compiled into microcode (analogously to FORTRAN and machine code). The arguments were the standard ones for higher-level languages. When he finished and asked for opinions, the consensus seemed to be that efficiency problems would make such languages impractical. Of course, this was the strongest argument against FORTRAN in its early days also, and time has shown the argument to be hollow. Nevertheless, the two cases are not quite equivalent and so the issue was not resolved.

*but see Briley, P. E., "Picoprogramming: A New Approach to Internal Computer Control", AFIPS 1965 Fall Joint Computer Conference, pp. 193-93; May-June 1966.

The topic of writable control stores prompted a vigorous floor debate on the advisability of allowing users to modify the control store - a "freedom vs. security" question, in this writer's opinion. Not surprisingly, manufacturers expressed a reluctance to offer this facility, envisioning a documentation and maintenance nightmare. One even expressed concern that users would "hang themselves" (perhaps with too much rope memory?). It seemed clear from the discussion that writable control stores are destined to come very much into demand, and that the technology governing their proper use will have to advance rapidly.

Joseph E. Sullivan
Information Processing

TRENDS

Ascher Opler - Died February 24, 1969

The Newsletter Editor wishes to pass on the news of the unfortunate death of one of our colleagues, Mr. Ascher Opler. Mr. Opler died in New York City suddenly the 24th of February 1969.

Mr. Opler was a consultant to the IBM Director of Research. He joined IBM in 1967 after nine years with Computer Usage Company and eleven years with Dow Chemical.

Mr. Opler had been associated with various aspects of the computing field since 1947. His activities included computer applications, development of automatic programming systems, and the publication of more than fifty-five articles on various aspects of computing. Among these were several dealing specifically with Microprogramming in which he coined the term "firmware".

Mr. Opler was also an Associate Editor of the Journal of the ACM.

SPECTRA 70/60 ANNOUNCED

The SPECTRA 70/60 was announced in March of this year and is now the top of the SPECTRA line. The SPECTRA operates on the same 145 instructions as the other SPECTRA processors. The interesting thing to note here is the use of the ROM (ROS) in the implementation of the 70/60. The previous top of the line, the 70/55, was not microprogrammed. This in itself laying to rest a commonly held opinion that microprogrammed architecture is suitable for slower processors only and that a large scale high speed system requires the use of standard sequential logic in its design - the characteristics of the 70/60 have been published as follows -

SPECTRA 70/60*

	Main Memory	(Main Store)
Cycle time		1 usec
Capacity		131 KB - 1,049 KB
Access		4 bytes
	Scratch Pad Memory	(Local Store)
Access Time		100 n sec
Capacity		128 words (32 bits)
	Read Only Memory	(ROS)
Access Time		330 n sec
Capacity		3072 X 72

* From Computer World 12 March 1969

CALL FOR PAPERS

The 1969 Workshop on Microprogramming will be held in Phoenix, Arizona, October 13 & 14, 1969. This, the 2nd Annual Workshop, is being co-sponsored by the ACM/SICMICRO and the IEEE/Computer Group/Central System Subcommittee. The expressed intention is to bring together technical people actively concerned with the use of microprogramming in the design or utilization of electronic digital processors.

Among areas of interest to be represented are:

- SYSTEM ARCHITECTURE
- LOGIC DESIGN
- HARD/SOFTWARE INTERFACE DESIGN
- THEORY
- SYSTEM ENGINEERING
- APPLICATION PROGRAMMING
- PEDAGOGICAL USES OF MICROPROGRAMMING
- LANGUAGE PROCESSORS
- MEMORY DESIGN AND IT'S RELATED TECHNOLOGY
- LSI-MICROPROGRAMMING RELATIONSHIPS

Requirements:

- Papers should be of current interest and related to the field of Microprogramming.
- Each contributor must arrange for necessary company or security clearances before submission.
- Each paper must include an abstract of not more than 200 words and not be more than 7500 words in length.
- Subjects selected for presentation do not require submission of a "finished paper", but the committee urges the participant to provide two copies for publication in the Workshop Summary.
- Abstracts must be received by 8 August 1969.

Authors will receive detailed instructions to be observed upon receipt of an initial abstract.

To receive detailed instructions relating to the admission criteria for attendance or the procedure for the submission of a paper, please send a post card to either of the chairmen listed below.

Program Co-Chairmen:

John R. Douglas (B-124)
SICMICRO Program Co-Chairman
General Electric Company
13430 North Black Canyon Highway
Phoenix, Arizona 85029

Dr. Bruce Briley
IEEE/Computer Group/
Central System Subcommittee
Bell Telephone Laboratory
Naperville, Illinois 60540