ANNOTATED MICROPROGRAMMING BIBLIOGRAPHY

Microprogramming Group

JULY 1970

Prepared for

DIRECTORATE OF PLANNING AND TECHNOLOGY
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

This document has been approved for public release and sale; its distribution is unlimited.
When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.
ANNOTATED MICROPROGRAMMING BIBLIOGRAPHY

Microprogramming Group

JULY 1970

Prepared for

DIRECTORATE OF PLANNING AND TECHNOLOGY
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 700A
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract F19(628)-68-C-0365
FOREWORD

This report was prepared by The MITRE Corporation, Bedford, Massachusetts, under Air Force Contract No. F19(628)-68-C-0365. It carries MITRE Project No. 700A. There is no Air Force System, Project or Task Number assigned. The report is a bibliography which provides an indexed, annotated list of publications related to the field of microprogramming. It covers the period from 1951, when the term was introduced, until the present time.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

Robert P. Savoy
Electronics Engineer
Development Engineering Division
Directorate of Planning & Technology
ABSTRACT

This bibliography provides an indexed, annotated list of publications related to the field of microprogramming, from the introduction of the term in 1951 to the present time.
ACKNOWLEDGEMENT

Documents for this bibliography were collected by Thomas Berschback and Nancy Anschuetz. A number of entries were obtained from the bibliography published by John Douglas, General Electric, Phoenix, in the SICMICRO Newsletter, Vol. 1, No. 2.

After initial review by Thomas Berschback, revisions were made by Robert W. Cornelli, Robert G. Curtis, Barbara J. Huberman and Codie S. Wells.

The KWIC index program used was prepared by William Amory.

Editorial control was exercised by Thomas Berschback throughout most of the preparation process.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>CONTENT</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTION I</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>SECTION II</td>
<td>BIBLIOGRAPHY</td>
<td>3</td>
</tr>
<tr>
<td>SECTION III</td>
<td>INDEX</td>
<td>51</td>
</tr>
</tbody>
</table>
SECTION I

INTRODUCTION

This bibliography provides an indexed, annotated list of publications related to the field of microprogramming, from the introduction of the term in 1951 to the present time.

This bibliography has been winnowed to exclude publications related only to the technological aspects of the memories, whether read-only or read-write, used to implement microprograms.

Section II is the annotated bibliography. Entries are organized alphabetically, by first author. Otherwise unidentified quoted sections are taken from the paper.

Section III contains a computer generated index, which provides access to bibliography entries by:

1. Key words in titles.
2. Authors' names.
3. Date of publication.
SECTION II

BIBLIOGRAPHY

(1) ALLEN-BABCOCK COMPUTING, INC., "Information outline: Remote-access
time-sharing system".

This sales brochure for ABC briefly describes their time-
sharing system. Their 360/50 "...has been modified by the attachment
of a second Read-only-store (ROS) which contains additional operation
codes...performs operations such as list searching, evaluation of
PL/I expressions, floating decimal arithmetic, and various utility
functions, all of which result in the fastest interpretive system in
existence."

(2) ALLEN, M. W.; PEARCEY, T.; PENNY, J. P.; ROSE, G. A. and SANDERSON,
J. G., "CIRRUS, An economical multiprogram computer with microprogram
control", IEEE Transactions on Electronic Computers, Vol. EC-12;
December 1963, pp. 663-671.

Machine: CIRRUS

CIRRUS, constructed in 1963 at the University of Adelaide,
Australia, is a general-purpose computer built for test purposes and
is not being produced. Some mention is made of microprogramming, but
only as a mechanism which allows certain operations to be handled
with more efficiency.
Machines: Thompson Ramo Wooldridge 133 (now the Bunker-Ramo 133), Packard-Bell 440, Collins 8401

"Microprogrammed computers basically differ from conventional computers in that the equivalent of a single instruction in a conventional computer is often a subroutine (interpreted instruction) in a microprogrammed computer." In such computers a program can be considered a calling sequence. The calling sequences of the machines listed above are described to show how the control sequences are simplified and more flexibility is allowed through the use of microprogramming. The cost of additional execution time is compared with the flexibility provided by microprogramming and the time saved in executing some complex instructions.

Machine: Interdata 3

This article presents several applications of Interdata 3 computers with special microprograms implemented to achieve significantly improved efficiency. The author claims that the use of read-only memories makes the use of small computers (like the Interdata 3) practical for application to process control problems.

"The authors suggest an approach to the use of algebraic languages, by means of hardware interpreter. A design is presented for the construction of a machine for the execution of a subset of FORTRAN consisting of arithmetic assignment, GO TO, computed GO TO, arithmetic IF, PAUSE, DO, CONTINUE, unformatted READ and WRITE, END and DIMENSION statements. Although such a machine has not been constructed, the authors make it clear that such construction is feasible. State diagrams are given for some of the most important circuits, and the hardware implementation is given for one of these circuits."


Machine: IBM 360/30

This article, originally in French and only roughly translated, addresses the design of a general-purpose or universal system to simulate any microprogrammed computer. As an example, a simulation of an IBM 360/30 is presented - a convenient choice because both the micromachine and the 360/30 contain 32 bit words. The authors note that the quality of the simulation is greatly enhanced by such matching where possible. The simulation of the 360/30 includes representations of general registers, instruction location counter and core memory, but not the I/O operations.
This article describes an approach for simplifying the control structure of a machine which utilizes the benefits of large-scale integration (LSI). "This evolutionary development has gone from conventional machine design first to a one-dimensional functional control partitioning (the register machine) and then to a two-dimensional elementary operation formatted control partitioning (the LIMAC machine)."

The two "dimensions" are: information transfer and data processing execution. Use of this structure "...leads naturally to the elementary operation (EO) format of control (also known as microprogramming)." In LIMAC (Large Integrated Monolithic Array Computer), a machine instruction is executed by performing a sequence of microinstructions. First, the operands are transferred into a function module, then another transfer operation passes the control word into a register to specify the execution.

The C-8401 is "...a stored program, intermediate scale digital computer...a microprogrammed data processor...communications oriented." The 1024 word, 36 bit micro-memory, though non-destructive, "...may be conveniently loaded under operator or program control."

**Machine: RCA Spectra 70/45**

The author, with startling brevity, discusses the concept of emulation, describes the RCA 301, the RCA Spectra 70/45 and its micromachine and the emulation process as applied to emulating the RCA 301.

"A micro-program simulation (i.e. emulation) is invariably more efficient than a macro-program simulation, since the simulation is being performed at the point of least difference between the computers."

BILLING, H. and HOPMANN, W., "Mikroprogramm-Steuerwerk", *Elektronische Rundschau* Nr. 10; 1955, pp. 349-359.

This paper, tutorial in nature, describes microprogramming as Wilkes defines it, and discusses read-only memory technology. In the original German.


"Simple computers exploiting the concepts of simulation and microprogramming are described. Logical rather than arithmetic microprogramming operations are employed for generality and greater simplicity. Design techniques are given for computers employing only multiple-bit time delays (i.e. no flip-flops or toggles are used)."

"The usefulness of developing a programming language for Ural-4 is justified. The following conditions are formulated which must be satisfied by the programming micro-language for Ural-4 and its translator: a substantial reduction of the programmer's work (compared to manual programming); the quality of programs constructed by the translator is not poorer than that of programs written by hand; the training of programmers with a good knowledge of Ural-4 should not exceed several days; the language admits of readjustment, extension, and inclusion in the automatic programming system; translation from other programming languages to the proposed one is simpler than translation to machine language; the work of programming the translator for this language should be minimal. The following basic properties of the microprogramming language for Ural-4 are pointed out: the instruction system coincides with that of Ural-4; the instructions consist of the operation code and certain names — alphanumeric words — used for locating the numbers taking part in the operation; symbolic addressing is used; the program can be divided into rows (connected sequences of instructions, joined by a single condition for execution); block programming is applied. An exact description of the language, its syntax, and the translation rules are not given."

G. V. in Cybernetics Abstracts, No. 9; 1967.

Machine: PB 440

The PB 440 is described as a microprogrammed computer with two applications: (1) as a control and arithmetic element in real-time systems, and (2) as a high performance calculator for general scientific purposes. The author states that a microprogram is much faster than a conventional subroutine.

Access to the read-only memory in the PB 440 is one microsecond, while the cycle time of the main memory is five microseconds.


Machine: PB 440

One of the original motivations for designing computers with a microprogramming control feature was to enable the programmer to direct the computer at a lower level of control than was possible with conventional, wired-logic computers. The logical structure of the PB 440, which fulfills this lower-level control requirement, is the topic of this paper.

The paper describes the general characteristics of the computer and also the different types of micro-instructions available. Included in the micro-machine is an automatic instruction counter, the P register.

An early time-saving characteristic, that of being able to initiate a read operation before finishing a restore operation in the main memory, is also introduced with the PB 440.

This paper discusses several possibilities for graceful degradation in adaptive computers. Graceful degradation occurs when a computer can continue to function, at some loss of accuracy or speed when a portion of the hardware is malfunctioning.

The author suggests the use of microprogramming: an alternate microprogram, not affected by a particular set of component failures can perform the failing function - at the cost of memory space.


Picoprogramming recognizes a "correspondence between the pulse-programming requirements of a control section and the capabilities of a memory element known as MYRA. A MYRA memory is a MYRIAperture ferrite disk which, when accessed, produces sequential trains of logic-level pulses upon 64 or more otherwise isolated wires."

The main advantage of picoprogramming is that no clock is needed because each MYRA disk, as it completes its switching, causes the next instruction to be obeyed.


"This paper discusses the specifications, programming implementation, and use of a design automation system that can be used in the development of any computer having read-only storage (ROS) controls.

This system, called the Controls Automation System (CAS), designed concurrently with the IBM System/360 was used in the development of Models 40, 50, 65 and 67. The current version of CAS is used by IBM to aid in the design of all systems with ROS controls.

The newly developed elements of this system are: 1) a programming language for describing the data flow of a machine; 2) a program for translating a variety of microprogramming languages into a common one, and 3) a set of computer programs enabling the machines operation to be simulated on an IBM 7090/1401 computer."

(From the paper)
Diagnostic procedures for the RCA Spectra 70/35 are implemented with a special diagnose instruction, which allows the execution of individual microinstructions ("elementary operations" or "functions") directly from main memory.

Two approaches to diagnostic procedures are discussed, both based on the use of the diagnose.

The first method involves testing at the instruction level until a problem occurs, then switching control over to the diagnose instruction. Major drawbacks to this approach are that: 1) sometimes the incorrect instruction cannot be isolated, and 2) the number of registers and the size of the instruction repertoire makes the number of possibilities to be tested prohibitively large.

The second method is a building block approach, in which testing is accomplished completely at the function level. For example, the transfer bus is tested after assuming the reliability of two registers. Then, if the transfer of information from one register to the other fails, the transfer bus should be at fault. The routine will continue if the transfer information is valid. The aim of the diagnostic routine is to isolate and test as few unknowns as possible at one time to better pinpoint a malfunction.

A concise paper on the advantages of microprogramming in the Spectra 70/35. The 70/35 elementary operations, which include register-to-register and variable operations, are discussed with an example given of the microprogram for the SS format MOVE instruction. This paper emphasizes the practicality of the read-only memory design method for both user and manufacturer.

Mr. Chu defines a microprogram as a series of binary numbers linked together. He has developed an ALGOL-type symbolic language to describe microprograms. This paper introduces the language by describing as an example a serial complement sequence using both normal computer techniques and microprogrammed control.

The difference between sequential logic control and microprogrammed control is (in this particular system) a trade-off between time and flexibility. Normal control in the example requires four elementary operations, microprogramming requires six. However, the latter allows flexibility because each bit in the instruction word can cause one micro-operation to be executed. Because of this opportunity for paralleled operations, only three instruction words are actually needed.

Two drawbacks appear in this specific microprogram. Considerable time can be lost in addressing the control memory; and the multiplicity of micro-operations would require very long control words. Trade-offs are possible, but at the cost of the full flexibility of the bit-per-operation concept.

A simulator has been constructed to test this higher-order language and a print-out is provided.


Machine: IBM 7950 System

E. D. Conroy defines microprogramming as the "direct control by the programmer of more than one echelon in the hardware hierarchy." Microprogramming allows the programmer direct control over the elementary operations.

Only the abstract and summary of the paper is presented.

This paper describes a computer organization in which only the more complex operations (for example, variable length add) exist as instructions. Each such (primary) instruction is followed by some microinstruction words (up to 256) which specify microinstructions (for example, load accumulator with zero) and the conditions which must exist in order for these instructions to be executed. If such a condition arises, the execution of the primary instruction is interrupted to permit execution of the microinstruction depending on the conditions.

In this brief summary, discussion of the technique is omitted.


Machine: IBM 360/85

This paper has as its purpose the introduction of the IBM 360/85, both in design and performance, and assumes knowledge of the IBM 360 system.

The first section deals with the design of the computer, and introduces a high-speed buffer called a cache. The cache provides for more storage space in the main memory by holding those portions of main storage that are currently in use.
The ideal computer, according to this article, is one that must be able, via microprogramming, to encompass a variety of instruction sets and provide an easy means of switching these sets. Such a system could have the capability of general emulation.

To attain this goal, this interim report studies the standardization of microprogramming techniques and attempts to incorporate these benefits: 1) ease of microprogram preparation and the ability to load a control memory with different microinstructions in the field, 2) designing compilers and check-out systems for the microprogram, 3) create better debugging devices for microprograms, and 4) create an "ideal" logic simulator which analyzes instruction sets before they are committed to hardware.

The theoretical microprocessor would have three major units, each able to function independently. They are an internal processor, a main memory controller, and an I/O controller. Each of these units would be controlled by the read-only control memory. A description of the read-only memory and the planned microinstruction format is also given.


Machine: Elbit 100

This paper describes the Elbit 100 computer. It argues the merits of a two-level memory computer, one level of which is the normal storage unit, while the other is a read-only memory. A speed increase due to the read-only memory occurs not from faster computing circuits, but from operational instructions built into the hardware for more efficient sequences of orders.

A good schematic diagram of a read-only memory control is included in the article. One memory is a standard random-access core store with a two microsecond cycle time while the other is a microprogrammable read-only store with 400 nanoseconds access time. The Elbit 100 has perhaps the simplest micro-machine of any existing computer.
This brief but excellent article provides a description of micro-instructions (the author terms these micro-orders), the information that each micro-instruction must contain, and some of the applications of microprogram control.

The three types of information required are: (1) operational, which controls the execution of a statement; (2) address information, which in some manner contains the address of the next micro-instruction; and (3) time information, which determines the processing time of the micro-instruction.

Under the heading of applications, the author's main point is that microprogram control can execute complex operations more efficiently by enabling the programmer to take into account the special properties of the algorithm, as well as using simpler circuits to generate the control circuit.

Finally, the author hypothesizes that two or more microprogram control computers can work together efficiently.

As in many of the better technical papers, the authors early and concisely state their aims:

It is the objective of this paper to briefly trace the history of the idea (microprogramming) and the difficulties involved with defining or implementing it. In doing this, we first consider the general control problem and instruction formats. Next, storage implementations of the control function are considered and a restricted definition of microprogramming is proposed. This is then evaluated from a technological, architectural and programming point of view. We hope to show that our (demanding) definition of microprogramming is now technologically feasible and attractive from systems consideration.

In addressing these objectives the authors have produced a worthwhile paper.

The development supporting the definition of microprogramming is clear and to the point. Consideration of technological, architectural, and programming implications of this early development is less successful, but still worthy of attention. On occasion the authors resort to designing when they should be stating requirements or categorizing potential properties, but their design sketches hit a nice level of complexity for their presentation purposes. Fired with enthusiasm, Messrs. Flynn and Maclaren are guilty of identifying and discussing only some of the problems and potentials of microprogramming (according to their definition, which assumes a read/write control store).

The problems of microprogramming on the level contemplated by the authors are those of programming. No observation is made that the control store becomes yet another level to be (dynamically) managed within the hierarchy; rather, specific ad hoc management techniques are presented. Similarly, deadlock potential in a multi-programmable microprogram is ignored, and protection problems are considered by design sketch rather than by requirement specification.
In perspective, however, the paper is of interest because of its structured, considered approach to some real problems, including some credible comments on the software aspects of microprogramming—a subject too often completely ignored in the literature.

The references are adequate and supportive, and the general form of presentation satisfactory (in spite of several typographical errors).


Starting with Wilkes' logical design of a control unit, this paper develops some variations and extensions. Specifically, it deals with problems concerning the conditions intervening at the end of a micro-order to determine the next one, with timing of micro-operations and also time-sharing and multiprogramming.


This article, written in August 1955, is an attempt to provide a rough outline of some conditions under which microprogramming might be a useful technique. Two methods of constructing a microprogrammed machine, a plugboard control and an internal relay setting control, are discussed in Section IV of the paper.

Efficiency is touted as the basis for a decision in favor of using micro-operations. The problems encountered in microprogramming are discussed in general from both an engineer's and a programmer's viewpoint.
In this paper a design is given for a program-modifiable control unit which does not require the usual high speed control store. A wired-in control memory contains microinstructions stored without repetitions. These are executed by means of words stored in core memory, each word containing the addresses of several micro-instructions plus looping information. The logic necessary to sequence these words is quite complicated. Furthermore, the user is restricted to the wired-in microinstructions.

"The author refers to the reviewer's expressed opinion that a program-modifiable microprogram system would probably not justify itself in practice, but suggests that in the area of non-numerical information processing of the 'production' type this may not be so. The case is not, however, argued in detail, and the author soon passes to a description of a proposed method of designing a program-modifiable control unit.

In the conventional microprogramming system a read-only memory is used to hold the program of micro-orders. In the proposed scheme the read-only memory contains the micro-orders arranged in a random order, each order used in the microprogram appearing once only. There is a secondary memory of normal erasable type containing words into which information about the microprogram is packed. These words are processed in order by the sequencing circuits, and each causes a sequence of micro-orders to be extracted from the read-only memory and executed.

The scheme is ingenious but complicated. The author realizes this, and he also draws attention to the other major disadvantage, namely, that the microprogrammer is restricted in his choice of micro-orders to those which are wired into the read-only memory. This might seriously prejudice the main object of the scheme, which is to get closer to ultimate efficiency by being able to choose an order code optimum for the application in hand."


"This paper is concerned with the no-man's land between, or common to, machines and languages; the land of soft hardware and hard software. We know the problem of finding a machine in terms of which programming languages can be defined; what of the problem of finding a language in terms of which machines can be defined? There is a need for a metalanguage in which no distinction need be made between hardware and software, between machine and language, so that the gap which exists in practice can be bridged in discussion.

Some confusion to the reader is likely on first reading owing to the fact that the word 'automaton' is incorrectly printed as 'automation' in a number of places."


Machine: μ-212

This set of class notes for "Engineering 212" presents the detailed design of a microprogrammed computer, the μ-212, which was created to highlight microprogramming aspects. Descriptions of the central processing unit, basic instruction set, and control unit (read-only) are listed. Three appendices are also provided:

1. Hardware drawings
2. Various microinstructions
3. Operation codes

The read-only memory has a 170 nanoseconds cycle time divided into three phases. Period one executes the current microinstruction and loads the control memory address register with the next microinstruction. Period two starts the reading of the new instruction from the address register, and period three places the instruction into the memory buffer register.

The authors claim the novelty of their read-only memory control design, but do not elaborate.
This paper introduces a computer designed jointly by Kyoto University and Tokyo Shibaura Electric Company Ltd. The system design, circuit elements and physical construction of the computer are briefly described. The order code of the machine is under the control of the programmer.

The fixed memory is unique in that it has been constructed with phototransistors. A punched card template is placed over an array of phototransistors and, by means of projected light, a path in a matrix is either opened or remains closed.

The output of the read-only memory is used to open and close gates (1 gate per bit organization). There are 256 eighty bit words in the ROM.

"The author defines a microprogrammed machine as one that has only those orders wired in that can be done in one machine cycle. Actual executed instructions are built up out of these basic operations. Generally, each 'macro-instruction' is executed as some sort of a hardware subroutine. The paper discusses the advantages of such machines, especially in the case where the user is allowed to design his own hardware subroutine. Of course, this technique is used in the hardware emulators of other machines used in IBM's System/360 machines."

The application of microprogramming to problem-oriented languages is described in terms of a simulated analog system on a digital computer. In the system described a problem defined by differential equations is drawn up in the form of an analog diagram. The system allows the machine assembly code to be freely mixed with the analog-oriented source language input in coding the supervisory and interrupt facilities.

"Quick, comprehensive error detection is a requisite of high reliability."

Hardware solutions are either too expensive or incomplete, while software solutions are both incomplete and result in severely slowed performance.

"If redundant software instructions are implemented as microinstructions, then the speed of checking is greatly increased."

"In this report, redundant algorithms and corresponding microprograms are presented and evaluated for detection of both transient and steady errors in parallel arithmetic and logical operations. The results are quite promising with respect to potential speed of performance, percentage of added hardware, and comprehensiveness of error detection."

Machine: IBM 360/40 (CP-40)

This paper discusses the use of the Associative Memory (AM) used to implement the Cambridge System, a modified 360/40. The Cambridge System allows up to 15 users to time share what appears to each user to be a standard, complete, but slow 360/40. The AM is used for automatic address translation, supporting a paging concept; it is controlled via microprogram.

The AM is functionally described, as are the controlling microprograms.


The author evaluates "...the contributions of stored logic to computer design...from the viewpoint of programming and applications."

He concludes that the techniques have not proved themselves, that "...the developers...found they had put together general purpose devices directly competitive with more conventionally organized computers in any given applications area."

"...Stored logic loses its significance when a stored logic computer is called upon to compile and execute FORTRAN, COBOL, NELIAC, JOVIAL, etc."

"...the future of stored logic as a design technique is somewhat cloudy."
This course is designed to introduce the student to an in-depth study of microprogramming for the IBM System 360 series. It describes the engineering techniques employed in microprogramming, but the primary concern is for the programmer's point of view.

Section I includes an introduction to read-only storage and control points. Section II contains ideas on the microprogram and control file memories. Section III is concerned with Controlled Automated System (CAS) logic diagrams.

This thesis is a comparison between a conventional computer, the UNIVAC 1830, and an experimental computer similar to the UNIVAC 1830 which uses a microprogrammed control unit.

The first section contains a general outline on the difference between conventional and microprogram control, followed by the test itself. The test results clearly show that the microprogrammed computer is faster; for example, the add operation under microprogramming takes 275 microseconds, while conventional means would take 1016 microseconds. This test, operated under hypothetical conditions, appears to contribute to the theory that microprogramming can contribute to a more efficient computer.
This chapter from Iverson's book defines microprogramming to include paper descriptions of a computer's machine language which could be used to bridge the gap between the computer design and the logical circuit designer. Iverson proceeds to describe the 7090 machine language in terms of more elementary operations.

This paper discusses the logical design of a binary, parallel, real-time computer. A description of the type of machine wanted is followed by the design decisions generated by the description. The contents of the micro-controlled logic generation are discussed, with reasons given for their incorporation into the scheme.
"This paper summarizes the principal characteristics and the advantages of the read-only memory (ROM) as a control element in the 4200/8200 processors. The functions of both the address-generator ROM and the arithmetic unit ROM in the Model 8200 are described. Bit steering is discussed as a decoding scheme used to reduce the size of the ROM in the Model 4200." (Author)

Described with facility poor even for a house journal, bit steering is an encoding scheme intended to provide "a method by which one could change the way ROM bits were interpreted without having to rewire the machine." The full flexibility achieved was not what might have been desired. Consider the number of $\mu$-operations which may be explicitly and concurrently mentioned "by (encoded) name" by $b$ bits. On one hand we may associate one bit for each of $b\mu$-operations, allowing concurrent mention of $b$ distinct $\mu$-operations. Alternatively, one might associate a $\mu$-operation with each of the $2^b$ states uniquely expressable by the $b$ bits, allowing concurrent mention of only one of the $2^b$ possible $\mu$-operations.

"Bit steering" chooses a middle ground. Given a field of $n + b$ bits, let the $n$ bits select ("steer to") one of $2^n$ sets of association maps for the $b$ bits. Each association map pairs each of the $b$ bits to one $\mu$-operation. One has the potential of addressing up to $2^n \times b\mu$-operations, $b$ at a time. In practice, the association maps are not mutually exclusive.
In a fourth generation computer with a changeable control store, microprograms can be used to implement a variety of instruction sets. It then becomes practical to translate a higher level language into some intermediate language which efficiently and conveniently expresses the higher level language, and then interpret the intermediate language by means of a microprogram. Six candidates for this intermediate language are defined. These candidates are then compared and evaluated with respect to conciseness, complexity, dynamic capability and flexibility.

Microprogramming is described as "a technique for systematizing the design of the central sequencing and control in a digital computer". The incorporation of microprograms in groups of computers is then given, including the Spectra 70 systems and the IBM 360 systems.

The last section of the paper deals with the design features of microprogramming, such as a modifiable order code, a late design freeze, and the improvement of speed. Some reference is given to the microprogram's role in multiprogramming, but this is very sketchy.
This paper presents a factual, concise and in-depth explanation of the concept of a read-only memory, with particular reference to the read-only memories used in the IBM 360 systems.

The two technologies used at IBM for the read-only storage are: (1) transformer read-only storage (TROS), and (2) capacitor read-only storage (CROS). The fact that these are permanent, but easily replaceable, devices is their advantage over conventional computer devices.

The functions of the ROS program are explained, along with their relationship to the macro-instructions they are executing. The distinction between "interrupt" and "trap" is made explicit. "A Trap is a hardware-initiated asynchronous transfer to some other point in the Microprogram. An Interrupt is a ROS-initiated, asynchronous transfer to some other point in the Macroprogram."
MATSUSHITA, S., "A microprogrammed communication control unit, the TOSBAC DN-231", Preprints of papers presented at IFIP Congress 68, Edinburgh; Booklet D: Hardware I, pp. D108-D112.

Machine: TOSBAC 3400

The TOSBAC DN-231 is a microprogrammed communication control unit designed to provide more efficient passage of I/O information to and from a general computer system. It contains a multiplexor channel controlled by microprograms.

This article provides basic reasons for using microprogramming: 1) decreased software overhead, 2) the flexibility needed to incorporate new control methods and new message formats without changing hardware, 3) easier maintenance, and 4) the ability to use special functions to save trivial software tasks.

A number of special functions are realized by microprograms. One such function provides for automatic data-chaining, concatenating buffers automatically to accept data streams. Another instruction, called POL 0, polls all the lines in a designated range. POL 0 polls only those lines used more frequently. By combining these two instructions, using POL 0 only occasionally, the user can decrease overall polling time.

This article, without going into great detail, points out some of the potential benefits of microprogramming a communication channel.

The paper is primarily a description of the SCC 4700, "...a 16-bit general purpose digital computer that features a microprogrammed control section. The paper gives the details of the microcode sequences. The paper also gives a few of the authors' opinions of the microprogramming technique for computer control sections."

The authors conclude that "microprogramming is not a panacea...microprogramming must be considered as simply another way by which computers can be implemented."


Machines: IBM 360/30 and 1401

This paper describes the hardware available for the IBM 360/30 which makes the emulation of an IBM 1401 possible. Special consideration is given to the read-only memory which contains the microprogram to do the emulation. The flexibility allowed through microprogramming in reference to the instruction counter, sense switches and stop routine is discussed, and the author concludes that the inherent flexibility of the read-only storage allows the emulation of an IBM 1401.
Machine: The TRW-133 (BR-133)

"This is a very clear and concise description of a small general-purpose machine that was built with a fairly large set of primitive orders that can be combined into subroutines by the user, and are executed under the control of a sequence of interpreted macro-orders.

The author, in addition to his clear expression, is to be congratulated on his restraint in avoiding the term 'micro-programming'.

The machine was designed for shipboard NTDS usage, at far greater speed, reliability and I/O processing than that of its predecessors designed by the same company. With the exception of the interpretive execution of micro-orders, its memory, logic, and registers are conventional."

Herbert M. Teager, Cambridge, Mass., Computing Reviews, Vol. 6, No. 2; March-April 1965, #7440.

Use of microprogrammed control units is suggested for interfacing experimental apparatus or computer peripheral devices with a computer. "...conventional design process is straightforward, it has the inherent disadvantage that it must be repeated for each new configuration...a single design schema which was sufficiently general to accommodate a wide variety of computers and external equipment, and which could be quickly and easily particularized to meet the requirements for specific controllers." The internal design of such a controller is described in great depth.

The IBM 2841 disk, drum, data cell controller is cited as one example, and a microprogram for controlling PEPR, "...a computer-controlled CRT scanner used to automatically measure bubble chamber tracks which have been recorded on film", as another.


This paper discusses the design of an experimental character-processing computer for the interpretive execution of higher-level language programs.

The design specifies a 100 nanosecond instruction cycle for the microprogram instructions stored in a read-only memory, a fast memory for intermediate "scratch pad" computation, and input/output through a conventional computer coupled to a two microsecond main memory.

On this small, special-purpose computer, FORTRAN programs are compiled and executed. The big difference lies in the micro-programming of both operations.

In the first phase of compilation, statements are entered at an on-line typewriter. Key words (DO, DIMENSION, etc.) require only one key to be depressed - the microprogram echoes the full word. Statements are collected, error-checked and stored in digested form. The second phase completes the compilation process. Expressions are transformed into Reverse Polish form and space is allocated.

Program execution is controlled in the microprogram using an Instruction Stack and a Data Stack. The Instruction Stack provides program control, particularly useful in controlling nested DO statements. The Data Stack is used by the microprograms corresponding to each statement, particularly for function and expression evaluation.

A DEBUG switch allows on-line tracing of the execution of each statement, by the logging of its statement number.

Using a standard FORTRAN compiler on another machine, and hand coding the same machine, the authors conclude that their FORTRAN lies somewhere between the other two techniques in performance.
The first section of this article presents a general outlook on the concept of microprogramming, incorporating the work of M. V. Wilkes and H. T. Glantz.

Microprogramming is defined as "...the technique of designing control circuits of an electronic digital computer to formally interpret and execute a given set of machine instructions, Mi, as an equivalent set of sequences of micro-operations, elementary operations that can be executed in one pulse time."

The author gives an example of an adder in a hypothetical microprogram, explains how it works and shows why it would take less time than an ordinary add instruction. The other sections of the paper deal with rather unrelated specific problems.

This article stresses the use of the variable instruction concept to enhance reliability by means of controlled graceful degradation. When a component fails, the set of microprograms which implement the macro-operations dependent on that component are replaced with a new set not so dependent. Processing can then continue, perhaps less efficiently.

Much of the article is descriptive of the logic used to arrive at the VIC design; embedded in this is most of the design itself.

See also Spence, A. L., "Hardware and Software Interactions; A Machine Organization Solution, The Variable Instruction Computer".

"The micro-program technique produces desired external operations in a digital computing system by sequences of internal operations controlled by a stored program of internal orders called 'micro-orders'. The various 'micro-program' sequences are selected by a second level of command, called 'sequence command'. This evaluation of the technique consists of an investigation of the effect of use on hardware complexity, computing speed, and ease of use.

Investigation of design considerations indicated the choice of a system consisting of an arithmetic section designed to be controllable by micro-orders, a memory which stores the micro-order sequences, a second memory which stores the sequence commands, a third memory which stores the problem data, a unique modified index register system for memory address control, and appropriate buffer and control registers.

It is shown that the arithmetic section could be achieved with no more functional elements than the minimum required in any system for the selected arithmetic operations and computing times. No unusual requirements were found for the problem data and sequence memories other than the need for independent control. Investigation of the micro-program memory indicated that a capacity of $10^6$ bits with a speed comparable to the data memory would be desirable.

By investigation of a 'standard' operation and matrix multiplication program, it is found that the micro-program system was significantly faster than a 'typical' conventional system. However, it was about as fast as a conventional system modified to provide more effective subroutine operation with a resulting complexity which was comparable to that of the micro-program system...

(From the abstract)

Computing Reviews, Vol. 9, No. 1;

No legible copy of this thesis was available for review.
This paper deals with the future of computers. Mr. Opler projects that microprograms will be used more frequently. He predicts that fourth-generation hardware will use large-scale integration (LSI) logic specialized by microprogramming. Mr. Opler hypothesizes the speed of three elements in a fourth-generation computer: circuitry (LSI) takes 5 nanoseconds, the micromemory access takes 10 nanoseconds, and main memory access 400 nanoseconds. Therefore, microprogram interpretation of instructions is feasible. He also introduces the term "firmware" to designate microprograms in the control memory. He expects the most important potential of the microprogram to occur with the advent of a slow-write/fast-read memory. This would enable the programmer, via a special set of punch cards, for example, to change the contents of the micromemory, thus permitting him to define his own instruction sets.

Among other topics, the effects of microprogramming and its potential are examined. Designers have begun to study it in depth because several micro-instructions of 10-100 nanoseconds each can be executed in less time than access to the main memory (500-2000 nanoseconds).

The concept of virtual memory with peripheral devices containing more storage is discussed in relation to microprogramming by showing how micro-operations can decrease access time. This is a short but excellent summary of the advantages of microprogramming.

This article gives a description of the functions of microprogrammed control and makes a comparison between general-purpose and microprogrammed machines. It dwells on microprogramming from both the engineer's and programmer's viewpoint; and, without going into any depth, does a creditable job. It says that a microprogram may consist of: (1) a fixed sequence of words containing conditional micro-orders; (2) a variable sequence of words containing unconditional micro-orders; or (3) a combination of both.

A hypothetical comparison is drawn in performing a square root with conventional and microprogrammed instructions.


Machines: Standard Computer Corporation IC 3000, 4000, 7000

"The fourth generation computer is a systems-architecture-oriented device which stresses changes in structure, organization and utilization of computer techniques and programming skills."

Mr. Rakoczi thus defines his concept of the fourth generation computer. Since fifty per cent of the article is a sales brochure, the convenience of having the only fourth generation computer is obvious.

The division between third and fourth generation computers seems to lie in the use of microprogramming. The microprograms resident in third generation computers are a replacement for controlled logic. They are devices to simplify and organize design procedures and aid in maintenance problems. The microprogram in a fourth generation computer is in itself an inner computer, designed not only to help the engineer, but also the programmer. The inner computer controls all communication between the functional stations of the system, aiding in efficient subroutine handling.

Perhaps the most sweeping claim made is that the inner computer, along with a process called MINIFLOW, allows for complete emulation. Even the author admits that only one type of computer can be emulated at any given time. No examples are given of the time factor involved if more than one machine is to be emulated during the same process; and no discussion is given as to the effect when the emulated machine has a different length word from the micro-machine.

Machine: IBM 360/30

EMIR (acronym from the title) is a real-time microprogrammed collector of radar data, apparently implemented using an IBM 360/30 CPU. EMIR also includes communication circuits with the radar, and another, possibly remote, more powerful computer, which might perform functions such as trajectory plotting.

The author defines microprogramming by example of an add operation, and does so classically: each microinstruction, operating in a single cycle time, opens or closes gates which correspond to the bits in the microinstruction.

The 360/30 condenser memory is briefly described, as are the IBM programs for developing microprograms.

The author is most taken with the flexibility which would allow rapid adaptation to new developments in radar technology or usage.

The article appears in the original French.


Machine: Intergraphic

This paper describes a proposed microprogrammed interface computer to link many (from 13 to 50) general-purpose graphical terminals to a central processor. The digital computer is fast (3-5 nanoseconds for integrated circuitry and 100 nanoseconds access time to the read-only memory) and versatile because of its read-only memory.

The control section comprises a ROM, addressed from a "read" register, a micro-routine link register Q, decoding and timing logic, and a bit-wise programmable register F. One of the microprograms is a small Cartesian vector program with 28 micro-orders.
ROSEN, S., "Hardware design reflecting software requirements", AFIPS Conference Proceedings; FJCC 1968, pp. 1443-1449.

The author introduces microprogramming as one of a number of hardware developments (the first one he lists) which has "contributed to the development and the performance of multiprogramming and multiprocessor operating systems and compiling and translating systems."

A conventional definition is given, and the relation to simulators/emulators is pointed out.


Machine: IBM 360 Series

This short (two page) article describes the use of a special return register along with a counting address register that eliminates the need for an explicit micro-address for each new microinstruction. A description of the execution of a micro-operation in each of four modes follows. The modes are: normal, prepare-return, execute-return and jump. The only mode which requires an explicit micro-address is the jump mode, in which a field of the microinstruction is interpreted as the next micro-address.


A stored logic type of computer is one in which the basic logic expressed by micro-commands is implemented by hardware, and all computer instructions above this building block level are programmed. The authors discuss the merits of the stored logic computer in relation to ease of learning, ease of modification, speed, and versatility, but do not go into much depth.
Interpretation as a technique leads to performance losses of a factor of 10 to 15. The stored logic concept, say the authors, offers "interpretation without deceleration".

"One may consider computer instruction types as rungs on an 'abstraction ladder' rising from machine language toward natural human language, with distance from the bottom denoting level of abstraction with respect to the computer logic.

The bottom rung of the instruction ladder is held by the primary building-blocks of all types of computer instruction, namely micro-commands which cause the machine to open gates, trigger flip-flops, and perform other useful circuit functions.

In the Stored Logic concept, much of the wired logic which mechanizes the machine-language instructions customarily found in an instruction list is disregarded, and only the basic logic expressed by the micro-commands is implemented by hardware. The logic for any instructions above the micro-command level is stored in the computer memory in the form of a micro-program, i.e., a sequence of micro-commands. Hence, in the Stored Logic concept, any type of instruction can be made up out of micro-commands and stored in memory access as a macro-instruction. Thus, conventional machine language differs from an interpretive system only by a degree of abstraction..."
This paper, written during the embryonic stages of the VIC computer development, contains the elementary concepts of the microprogramming involved in it. The high speed memory, with a "scratch-pad" section, consists of two modules, each containing 256 x 38 bit words. On Page 5 is an excellent example of how a macroinstruction (add) is carried out by a sequence of micro-orders. Many reasons for the use of microprogramming are given including flexibility, utility and speed.

See also Miller, E. H., "Reliability Aspects of the Variable Instruction Computer".

This paper discusses the system implementations of the 360 series and measures performances in four areas: 1) main storage, 2) central processing unit, registers and data paths, 3) sequence control, and 4) input-output channels. Each of these is then further categorized into three factors: basic speed, size, and degree of simultaneity.

The read-only storage control is discussed under the major heading of sequence control. Besides listing certain characteristics of the read-only storage in the individual models in the 360 series, the author states, "This microprogram control...is the only method known by which an extensive instruction set may be economically realized in a small system." The IBM 360/60, 62, is presented as proof of this. Originally designed with conventional control logic, the IBM 360/60, 62, was altered for microprogramming control when its instruction set required too many circuit modules.

Written in the first stages of the evolution of microprogramming, this paper first overviews the original view of the concept, then questions the choices of order code that could be built into the microprogram.

Most of the article expounds certain advantages that seemed to provide a raison d'être for microprogramming. These include late design freeze, easier and faster floating-point arithmetic, and more efficient control methods.

This paper is interesting from the standpoint of examining the early endeavours into the field of microprogramming; but it does not introduce the reader into any in-depth concepts.


"The paper examines the possibilities of utilizing the micro-subprograms...to reduce the number of micro-orders and to increase the flexibility..."

"In new computers there are introduced complex operations which result from the nonarithmetical applications of these, from the automation problems of programming, multiprogramming, etc."

"In this condition of complicating the microprogram, formal synthesis methods must be used and the microprogram minimization becomes essential for design."

Machine: IBM 360 Systems

"This paper describes the kind of microprogram control that has been used in several models of System/360. A microprogramming language, as well as some of the main techniques used in "assembling" and testing microprograms, are discussed. Applications of micro-program control to the design of emulators, to compatibility features, and to special modifications are summarized."

(From the paper)


"This paper describes the use of a multiaperture ferrite device called MYRA, and the use of picoprogramming to build a 16-bit, one microsecond integrated circuit computer. Picoprogramming is described as conceptually comparable to microprogramming but without the need for decoding logic. The MYRA provides the means for picoprogramming.

Organization of the computer, implementation of a micro-instruction, operation of the MYRA, and the advantages and disadvantages of the technique are presented.

The paper is complete and well written."

H. Marks, Canoga Park, California, Computing Reviews, Vol. 9, No. 6; June 1968, #14,585.
This article discusses the practical limitations of the dissection of macro-instructions into more elementary operations. The ZEBRA computer is used to test this dissection.

The argument that construction of the micro-code is too laborious for the programmer (van der Poel's concept of microprogramming is slightly different from Professor Wilkes' -- he sees it as an easily adjustable elementary machine language) is outweighed by its efficiency, especially in the area of automatic programming.

Mr. van der Poel also defines two concepts: trickology and underwater programming. The generation of pieces of coding in fast registers which are subsequently executed is called underwater programming. Considerable ingenuity is often required to devise the macro-instructions, giving rise to the name trickology for the art of using this tricky programming.

Three-fourths of this article is a description of ZEBRA. The examples and micro-code used are also from this computer. The article may be of some interest to those designing micromachines, or those actively (!) concerned with coding for the ZEBRA computer.

"Basic concepts of microprogrammed control design, representation of current design practice, and selected design variations are examined as sources of informative value to the logic designer."

(From the paper)

Machine: RCA Spectra 70/45

MIDAS, the Microprogramming Design Aid System, describes a program which performs much of the routine work necessary in developing a useful, effective microprogram.

Designing a microprogram requires "that a set of micro-orders be combined into a set of elementary operations in such a way that 1) the required function is in fact performed, and 2) the execution time is minimized." MIDAS allows the programmer to concentrate on this creative task by permitting him to "see" his microprogram and check it by a simulation process.

MIDAS includes an assembler, a simulator, a program to punch cards by which the read-only memory can be built, and a system to aid in debugging.


Machine: IBM System/360 Model 30

This article describes an experimental processing system for a higher level language, EULER. The language is processed in two stages. First a microprogrammed compiler translates it into a Reverse Polish String Language. Then this language is interpreted by a microprogram. The experiment was undertaken to show that microprogramming can be used to create more efficient systems than conventional programming systems. This system is predicted to be at least ten times faster than the execution of an ordinary (efficient) machine code compilation.

In this course, Mr. Weber introduces the microprogram level language by analogy - as ALGOL is to machine language, so machine language is to a microlanguage.

Using several of the System 360 micromachines as examples, Mr. Weber weaves through the classical and later definitions of microprogramming, the types of read-only memory that are used and the possibilities of read-write memories. The 360/30 (2030) and 360/20 Submod 5 (2020-5) are examined in detail.

He also points out that "...a simulator, an interpreter, and the machine hardware performed exactly the same function, namely, to interpret string code. Why was it necessary to invent different names for simulator and interpreter?" "...emulators were announced. Again, a new word for an interpreting mechanism!"

Also, "the ideal processing system for a higher level language would therefore consist of a translator and an interpreter. Both the translator and the interpreter can be written in a microprogram language."
This paper contains the original explanation and definition of the term microprogramming. A micro-operation is one of a series of steps. These steps may include transfers from the store to control or arithmetical registers and transfers from one register to another. The micro-operations can be generated by pulses. These pulses enter a decoding tree, and then are routed to two matrices, one of which executes an instruction, the other sets up the next instruction to be performed.

When introduced, microprogramming's main advantage seemed to Wilkes to be the increase in the degree of reliability in a machine. Reliability had a direct relationship to 1) the amount of equipment of a computer, 2) its complexity, and 3) the degree of repetition of units. Since microprograms can perform many operations with the same set of logic circuitry, it would appear that microprogramming did increase the reliability of a machine.

Since this was the original paper on the subject, the problems of programmer instruction, design, and time consumed are not discussed in detail.

This article, written in 1958, gives a very brief outline on the development of microprogramming, including its advantages and potential. A microprogram is defined as "the list of suboperations or micro-operations required to execute all the orders in that order code".

The one outstanding advantage of microprograms is that changes in the order code can be incorporated at a very late date in the building of a computer.

The other new concept of the erasable or "scratch-pad" memory which allows the programmer to invent his own order code is also mentioned.
WILKES, M. V., "The growth of interest in microprogramming", ACM Workshop on Microprogramming, sponsored by the Association for Computing Machinery in cooperation with The MITRE Corporation, October 7-8, 1968.

Tracing the evolution of microprogramming from its conception in 1951 to the present, M. V. Wilkes examines its original and current uses and theorizes on further potential development.

A helpful explanation of the two current and different definitions of microprogramming are included. The concept of stored logic (a read-write control memory), a technique which allows the programmer to create his own microprograms, is also mentioned.

A history of microprogrammed computers is compiled, with brief comments provided on each referenced article.

Tucker's reasons for the use of microprogramming are summarized. It provides: 1) an economical way for smaller computers to have large instruction sets, 2) an aid to efficient maintenance, 3) the possibility of emulation, and 4) a gain in flexibility for both the programmer and designer.

The current state of the art is discussed in terms of further developments in emulation, software support, and the interpretation of higher-level programming languages.


"The literature is surveyed beginning with the first paper published in 1951. At that time microprogramming was proposed primarily as a means for designing the control unit of an otherwise conventional digital computer, although the possible use of a read/write control memory was noted. The survey reveals the way in which interest has successively developed in the following aspects of the subject: stored logic, the application of microprogramming to the design of a range of computers, emulation, microprogramming in support of software, and read/write control memories. The bibliography includes 55 papers."

(From the paper)

This article, dated November 18, 1952, describes in clear, concise terminology the structure and advantages of a microprogrammed computer.

Building on his original paper which introduced the concept of microprogramming in 1951, Professor Wilkes defines microprogramming as a method of designing the control circuits of a machine to be logical and yet enabling the order code to be alterable without ad hoc alterations to the circuits. The concept of a micro-control unit is discussed. Branching within the microprogram is shown to be accomplished by the setting of two flip-flops.
### SECTION III

#### KEY INDEX LISTING

<table>
<thead>
<tr>
<th>TEXT</th>
<th>KEYWORD AND TEXT</th>
<th>INDEX</th>
</tr>
</thead>
</table>

**25 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> C**

**5 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> -**

**1 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> --**

**177 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**34 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> A**

**MCGEE, W & PETERSON, H** **MICROPROGRAMMING FOR DATA ACQUISITION AND CONTROL** **1965**

**BREUER, M** **ADAPTIVE COMPUTERS** **1967**

**VOLLRECHT, J** **MIDAS: MICROPROGRAMMING DESIGN AID SYSTEM FOR RCA SPECTRA 70/45** **1968**

**7 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> AL**

**CEESS TIME-SHARING SYSTEM**

**PROGRAM COMPUTER WITH MICROPROGRAM CONTROL** **1963**

**4 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> AN**

**8 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> AND**

**HILLYER** **STORED LOGIC PROGRAMMING AND APPLICATIONS** **1964**

**MEILITZ, H; LEVY, S; LINHARDT, R; MILLER, H** **SYSTEM ARCHITECTURE FOR LARGE SCALE INTEGRATION** **1967**

**LEVY, L** **STATE OF THE ART OF MICROPROGRAMMING** **1965**

**HORRIS, J** **INVESTIGATION OF MICROPROGRAMMING** **1965**

**CONTI, G; GIBSON, R; PETKOWSKY, S** **STRUCTURAL ASPECTS OF THE SYSTEM/360, MODEL 65** **1968**

**MILLER, R** **RELIABILITY ASPECTS OF THE VARIABLE INSTRUCTION COMPUTER** **1967**

**HERNES, S; ET AL** **CONTROL STORAGE USE IN IMPLEMENTING AN ASSOCIATIVE PROCESSOR FOR A TIME-SHARED COMPUTER** **1968**

**HELLEMA, L; ET AL** **CONTROL STORAGE USE IN IMPLEMENTING AN ASSOCIATIVE PROCESSOR FOR A TIME-SHARED COMPUTER** **1968**

**CALHOUN, R** **DIAGNOSTICS AT THE MICROPROGRAMMING LEVEL** **1969**

**WILKES, M** **THE BEST WAY TO DESIGN AN AUTOMATIC CALCULATING MACHINE** **1951**

**JERGINIAN, H; CARTER, W; CRAMER, J; & WELLS, G** **THE CONTROLS AUTOMATION SYSTEM** **1965**

**4 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**1967**

**MACHINE** **1967**

**STUDY ORIENTED TO COMPUTERS SIMULATION** **1968**

**E FOR LARGE SCALE INTEGRATION** **1967**

**RAILLY, S** **FASTER COMPUTER CONTROL WITH A READ-ONLY MEMORY** **1965**

**BASKOW, T; SASSON, A & KRONFELD, A** **SYSTEM DESIGN OF A FORTRAN LANGUAGE** **1965**

**RAI, B; FERRELL, R; & HUOT, P** **MULTIMACHINE UNIVERSAL STRUCTURE** **1966**

**BECK, L & KEELEY, T** **THE C-8401 DATA PROCESSOR** **1964**

**BILTON, M; LEVY, S; LINHARDT, R & MILLER, H** **SYSTEM ARCHITECTURE** **1965**

**BLANKENBACHER, J** **THE BEST WAY TO DESIGN AN AUTOMATIC CALCULATING MACHINE** **1951**

**10 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**17 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**18 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**23 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**37 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**38 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**45 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**52 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**77 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**

**133 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> "**
<table>
<thead>
<tr>
<th>TEXT</th>
<th>KEYWORD AND TEXT</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ING AND THE DESIGN OF THE CONTROL CIRCUITS IN AN ELECTRONIC GRAMMING THE SPECTRA 70/35</td>
<td>DIGITAL COMPUTER = 1953WILKES,M E STRINGER,J = MICROPROGRAMMING</td>
<td>84</td>
</tr>
<tr>
<td>DIGITAL COMPUTER = 1962 CAMPBELL,C &amp; NEILSON,D = MICROPROGRAMMING</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>YANOVI-YARoslAVSkiY, L E TIMOFEEV,A = MICROPROGRAM Control FOR DIGITAL Computers = 1962</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>DIGITAL COMPUTING SYSTEMS = 1958 MORRISON,G = INVESTIG</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>OPLER,A = NEW DIRECTIONS IN SOFTWARE 1960-1966</td>
<td>HATSUMITA,S</td>
<td>59</td>
</tr>
<tr>
<td>MELBOURNE,A &amp; PUGMIRE,J = A SMALL COMPUTER FOR THE DIRECT PROCESSING OF FORTAN STATEMENTS = 1965</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>A MICROPROGRAMMED COMMUNICATION CONTROL UNIT, THE TOSBAK</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>BENJAMIN,R = THE SPECTRA 70/45</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>GREEN,J = MICROPROGRAMMING, EMULATORS AND PROGRAMMING LANGUAGES = 1966</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

7 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> ET

WEBER,H = A MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODELS 30 = 1967 | 78 |

HARRAND,Y = EVOLUTION OF MICROPROGRAMMING CONCEPTS = 1963 | 34 |

MCGEER,M & PETERSUN,M = MICROPROGRAM CONTROL FOR THE EXPERIMENTAL SCIENCES = 1965 | 51 |

REBOLOT,M = EXTRACTEUR MICROPROGRAMME D'INFORMATION RADAR = 1968 | 62 |

BAILEY,S = FASTER COMPUTER CONTROL WITH A READ-ONLY MEMORY = 1967 | 4 |

MCCORMACK,M = SCHUMAN,T & WOICACK,K = 1401 COMPATIBILITY FEATURE ON THE IBM SYSTEM/360 MODEL 30 = 1965 | 49 |

ORIENTED TO COMPUTERS SIMULATION = 1968 | 6 |

COMPUTER - A MICROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY = 1962 | 33 |

YAMAUCHI,H ET AL = THE KT PILOT C | 33 |

COMPUTER - A MICROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY = 1962 | 33 |

MATSUSHITA,S ET AL = THE KT PILOT C | 33 |

FLYNN,M C MACLAREN,M = MICROPROGRAMMING REVISITED = 1967 | 27 |

16 ENTRIES WERE SUPPRESSED FOR KEYWORD --------> FOR

BASHKOW,T; SASSON,A & KRONFELD,A = SYSTEM DESIGN OF A FORTAN MACHINE = 1967 | 5 |

PUGMIRE,J = A SMALL COMPUTER FOR THE DIRECT PROCESSING OF FORTAN STATEMENTS = 1965 | 5 |


OPLER,A = FOURTH-GENERATION SOFTWARE: THE REALIZATION = 1967 | 58 |

TARY STRUCTURE = 1960 | 42 |


GERACE,G = MICROPROGRAMMED CONTROL FOR COMPUTING SYSTEMS = 1 | 28 |

CONTICI,C; GIBSON,D E PITKONSKY,S = STRUCTURAL ASPECTS OF THE SYSTEM/36 | 23 |

GLINTZ,H = A NOTE ON MICROPROGRAMMING = 1956 | 29 |

MED CONTROL UNITS = 1962 | 63 |

ROSE,G = INTERGRAPHIC, A MICROPROGRAMMED GRAPHICAL-INTERFACE COMPUTER = 1967 | 30 |

GRASSELLI,A = THE DESIGN OF PROGRAM-MODIFIABLE MICRO-PROGRAM | 31 |

GREEN,J = MICROPROGRAMMING, EMULATORS AND PROGRAMMING LANGUAGES = 1969 | 82 |

WILKES,M = THE GROWTH OF INTEREST IN MICROPROGRAMMING = 1968 | 83 |

GREENBERG,M & WEGREBERG,F = A MICROPROGRAMMED COMPUTER = 1968 | 32 |

WILKES,M = THE GROWTH OF INTEREST IN MICROPROGRAMMING: A LITERATURE SURVEY = 1969 | 33 |

GRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY = 1962 HAGIHARA,M; AMO,K ET AL = THE KT PILOT COMPUTER - A MICROPROGRAMMING = 1966 | 33 |
<table>
<thead>
<tr>
<th>TEXT</th>
<th>KEYWORD</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLUTION, THE VARIABLE INSTRUCTION COMPUTER &quot;1966&quot;</td>
<td>SPENCE, A</td>
<td>69</td>
</tr>
<tr>
<td>ROSEN, S</td>
<td>HARDWARE AND SOFTWARE INTERACTIONS: A MACHINE ORGANIZATION 5</td>
<td>64</td>
</tr>
<tr>
<td>HARDWARE DESIGN REFLECTING SOFTWARE REQUIREMENTS &quot;1966&quot;</td>
<td>HARRANDY</td>
<td>34</td>
</tr>
<tr>
<td>ASSOCIATIVE PROCESSOR FOR A TIME-SHARED PROCESSOR &quot;1968&quot;</td>
<td>MEGGITT, J</td>
<td>35</td>
</tr>
<tr>
<td>CHU, Y</td>
<td>A CHARACTER COMPUTER FOR A HIGHER-ORDER LANGUAGE FOR DESCRIBING MICROPROGRAMMED COMPUTE</td>
<td>53</td>
</tr>
<tr>
<td>ASSOCIATIVE PROCESSOR FOR A TIME-SHARED PROCESSOR &quot;1968&quot;</td>
<td>BILLING, G</td>
<td>20</td>
</tr>
<tr>
<td>BILLING, G</td>
<td>A MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D</td>
<td>38</td>
</tr>
<tr>
<td>SCHANSMAN, T</td>
<td>1401 COMPATIBILITY FEATURE ON THE IBM SYSTEM/360 MODEL 3D</td>
<td>39</td>
</tr>
<tr>
<td>WOMACK, K</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D</td>
<td>49</td>
</tr>
<tr>
<td>WEBER, H</td>
<td>THE STRUCTURE OF SYSTEM/360: PART II: SYSTEM IMPLEMENTATIONS</td>
<td>50</td>
</tr>
<tr>
<td>STEVENS, W</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D</td>
<td>78</td>
</tr>
<tr>
<td>ESSOR, W</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D</td>
<td>78</td>
</tr>
<tr>
<td>ESSOR, W</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>78</td>
</tr>
<tr>
<td>PATZER, W</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>78</td>
</tr>
<tr>
<td>ALLEN-BABCOCK COMPUTING, INC</td>
<td>INFORMATION OUTLINE: REMOTE-ACCESS TIME-SHARING SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>REBOULT, M</td>
<td>&quot;EXTRACTED MICROPROGRAMME&quot;</td>
<td>62</td>
</tr>
<tr>
<td>OSE COMPUTER &quot;1966&quot;</td>
<td>&quot;INTEGRATION&quot;</td>
<td>40</td>
</tr>
<tr>
<td>ROSE, G</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>44</td>
</tr>
<tr>
<td>LINNARD, R</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>8</td>
</tr>
<tr>
<td>MILLER, W</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>8</td>
</tr>
<tr>
<td>LAWSON, J</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>INSTRUCTION COMPUTER &quot;1966&quot;</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>MILKES, M</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>ROSE, G</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>TROL OF DIGITAL COMPUTING SYSTEMS &quot;1958&quot;</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>MORRISON, G</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>OLLEO COMPUTER WITH ELEMENTARY STRUCTURE &quot;1960&quot;</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>BECK, L</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>KEELER, F</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>NTROL LOGIC &quot;1968&quot;</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>BASHKOW, T</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>KOCHEN, G</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>RANSIS, T</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>YAMAUCHI, H</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>RANSIS, T</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>RANSIS, T</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
<tr>
<td>IVERSO, K</td>
<td>&quot;MICROPROGRAMMED IMPLEMENTATION OF EULER ON IBM SYSTEM/360 MODEL 3D&quot;</td>
<td>69</td>
</tr>
</tbody>
</table>

B ENTRIES WERE SUFFED FOR KEYWORD -------> IN
<table>
<thead>
<tr>
<th>TEXT</th>
<th>KEYWORD AND TEXT</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHU, Y.</td>
<td>A HIGHER-LEVEL LANGUAGE FOR DESCRIBING MICROPROGRAMMED COMPUTERS</td>
<td>1968</td>
</tr>
<tr>
<td>GREEN, J.</td>
<td>A CHARACTER COMPUTER FOR HIGH LEVEL LANGUAGE INTERPRETATION</td>
<td>1964</td>
</tr>
<tr>
<td>HAW</td>
<td>FLETCHER, V.</td>
<td>MICROPROGRAMMED COMPUTER FOR HIGH LEVEL LANGUAGE INTERPRETATION</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>SYSTEM ARCHITECTURE FOR LARGE SCALE INTEGRATION</td>
<td>1967</td>
</tr>
<tr>
<td>BEELITZ, H.</td>
<td>DIAGNOSTICS AT THE MICROPROGRAMMING LEVEL</td>
<td>1969</td>
</tr>
<tr>
<td>MEGGITT, J.</td>
<td>A CHARACTER COMPUTER FOR HIGH LEVEL LANGUAGE INTERPRETATION</td>
<td>1964</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>STATE OF THE ART OF MICROPROGRAMMING</td>
<td>1965</td>
</tr>
<tr>
<td>WILKES, M.</td>
<td>THE GROWTH OF INTEREST IN MICROPROGRAMMING: A LITERATURE SURVEY</td>
<td>1969</td>
</tr>
<tr>
<td>LIVINGSTON, J.</td>
<td>A SYSTEM/360 PROFILE -- READ-ONLY STORAGE</td>
<td>1969</td>
</tr>
<tr>
<td>AYDIN, L.</td>
<td>MICROPROGRAMMING AND STORED LOGIC</td>
<td>1964</td>
</tr>
<tr>
<td>KLEIN</td>
<td>S &amp; SCHWATZ, M.</td>
<td>MODEL 4200 - 8200 READ-ONLY MEMORY CONTROL</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>STORED LOGIC COMPUTING</td>
<td>1961</td>
</tr>
<tr>
<td>WILKES, M.</td>
<td>THE BEST WAY TO DESIGN AN AUTOMATIC CALCULATING MACHINE</td>
<td>1960</td>
</tr>
<tr>
<td>JONES, J.</td>
<td>A SYSTEM ARCHITECTURE FOR A FORTRAN COMPUTER</td>
<td>1963</td>
</tr>
<tr>
<td>FLYNN, M.</td>
<td>A HARDWARE AND SOFTWARE INTERACTION A STUDY</td>
<td>1965</td>
</tr>
<tr>
<td>MACLEAN, R.</td>
<td>MICROPROGRAMMING REVISED</td>
<td>1967</td>
</tr>
<tr>
<td>THE TISSAC ON-231</td>
<td>1968</td>
<td></td>
</tr>
<tr>
<td>DODGE, P.</td>
<td>A COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY</td>
<td>1962</td>
</tr>
<tr>
<td>TUREN, P.</td>
<td>IN THE IBM SYSTEM/360 MODEL 30</td>
<td>1965</td>
</tr>
<tr>
<td>NTAL SCIENTIFIC AND TECHNICAL</td>
<td>1965</td>
<td></td>
</tr>
<tr>
<td>ERPELATIV</td>
<td>1964</td>
<td></td>
</tr>
<tr>
<td>JELSMA, J.</td>
<td>A FORTRAN STATEMENTS</td>
<td>1965</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>A MICROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY</td>
<td>1962</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>A SMALL COMPUTER FOR THE DIRECT PROM</td>
<td>1964</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>A FAST COMPUTER CONTROL WITH A READ-ONLY MEMORY</td>
<td>1967</td>
</tr>
<tr>
<td>MILLER, H.</td>
<td>A MICROPROM PROGRAMMED COMPUTER</td>
<td>1968</td>
</tr>
<tr>
<td>MERCEC, L.</td>
<td>A MICROPROGRAMMING</td>
<td>1957</td>
</tr>
<tr>
<td>SLOBOJDANOK, V.</td>
<td>A METHOD OF MINIMIZING MICROPROGRAMS</td>
<td>1967</td>
</tr>
<tr>
<td>BURCHELL, J.</td>
<td>MICROPROGRAMMING: A LANGUAGE FOR URBAN DIGITAL COMPUTERS</td>
<td>1966</td>
</tr>
<tr>
<td>GRASSIELI, F.</td>
<td>A DESIGN OF PROGRAMMABLE MICROPROGRAMMED CONTROL UNITS</td>
<td>1962</td>
</tr>
<tr>
<td>MEERK, R.</td>
<td>MICROPROGRAMMING</td>
<td>1957</td>
</tr>
<tr>
<td>VALUER, R.</td>
<td>MICROPROGRAMMING AND TRICKOLY</td>
<td>1962</td>
</tr>
<tr>
<td>DODGE, P.</td>
<td>A MICRONSTRUCTURE</td>
<td>1961</td>
</tr>
</tbody>
</table>
SONI,D & PITKOWSKY S " STRUCTURAL ASPECTS OF THE SYSTEM 360 " MODEL 85 " 1968 CONTI,G; GIB

VA LASSIS,J " MODULAR COMPUTER DESIGN WITH MICROPROGRAMMED CONTROL " 1967 76

JE IN THE CONTROL OF DIGITAL COMPUTING SYSTEMS " 1962 MCRESSON,G " INVESTIGATION OF MICRO-PROGRAMMING AS A TECHNIQ

ALLEN, W PEARCEY, T PENNY, J ET AL " CIRRUS, ECONOMICAL MULTIPROGRAM COMPUTER WITH MICROPROGRAM CONTROL " 1963 2

KUSSE; GI SANDERSON, ET AL " CIRRUS, ECONOMICAL MULTIPROGRAM COMPUTER WITH MICROPROGRAM CONTROL " 1963 2

C NEILSON, D " MICROPROGRAMMING THE SPECTRA 10/35 " 1966 N A VARIABLE STRUCTURE DIGITAL COMPUTER " 1962 CAMPBELL, C

VARIABLE STRUCTURE DIGITAL COMPUTER " 1962 CAMPBELL, C & NEILSON, D " MICROPROGRAMMING THE SPECTRA 10/35 " 1966 N A

OPLER, A " NEW DIRECTIONS IN SOFTWARE 1960-1966 " 1966 59

DECISION SYSTEMS, INC " INTERIM STATUS REPORT NO. 1 " 1968 24

GLANZ, H " A NOTE ON MICROPROGRAMMING " 1956 29

BUCKINGHAM, B; CARTER, W; CRANFORD, W, & NIDELL, G " THE CONTROLS AUTOMATION SYSTEM " 1965 17

22 ENTRIES WERE SUPPRESSED FOR KEYWORD ------- OF

3 ENTRIES WERE SUPPRESSED FOR KEYWORD ------- ON

STRINGER, J " MICROPROGRAMMING AND THE CHOICE OF BOUTWELL, E & HUSKINSON, E " THE LOGICAL ORGANIZATION OF THE PB 44D MICROPROGRAMMABLE COMPUTER " 1963

986 SPENCE, A " HARDWARE AND SOFTWARE INTERACTIONS: A MACHINE ORGANIZATION SOLUTION. THE VARIABLE INSTRUCTION COMPUTER " 1

FERRE, J & HUGOT, P " MICROMACHINE UNIVERSAL STRUCTURE STUDY ALLEN, W; BACKUCK COMPUTING, INC " INFORMATION OUTLINE: REMOTE-ACCESS TIME-SHARING SYSTEM "

STEVENS, W " THE STRUCTURE OF SYSTEM 360; PART II - SYSTEM IMPLEMENTATIONS " 1964 PATZER, W & VANDLING, G " SYSTEM IMPLICATIONS OF MICROPROGRAM

MING " 1967 58

BOUTWELL, E " THE PB 44D COMPUTER " 1964 59

BOUTWELL, E & HUSKINSON, E " THE LOGICAL ORGANIZATION OF THE PB 44D MICROPROGRAMMABLE COMPUTER " 1963 14

CAN, D " MICROPROGRAM CONTROL " 1963 ALLEN, W; PEARCEY, T; PENNY, J ET AL " CIRRUS, ECONOMICAL MULTIPROGRAM COMPUTER WIT

CES " 1965 2

MCREE, W & PETERSON, H " MICROPROGRAM CONTROL FOR THE EXPERIMENTAL SCIENCE " 1965 51

" THE KT PILOT COMPUTER A MICROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY " 1962 HAGIWARA, H; AMO, K ET AL

" THE KT PILOT COMPUTER A MICROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY " 1962 YAMAUCHI, H ET AL

" THE KT PILOT COMPUTER A MICROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY " 1962 MATSHIWA, S ET AL

IBM " DATA PROCESSING: MICROPROGRAMMING PI COURSE "

VA LASSIS, J " MODULAR COMPUTER DESIGN WITH MICROPROGRAMMED CONTROL " 1967 39

SISTOR FIXED MEMORY " 1962 MATSUSHITA, S ET AL " THE KT PILOT COMPUTER A MICROPROGRAMMED COMPUTER WITH A PHOTOTRAN

SISTER FIXED MEMORY " 1962 HAGIWARA, H; AMO, K ET AL " THE KT PILOT COMPUTER A MICROPROGRAMMED COMPUTER WITH A PHOTOTRAN

SISTER FIXED MEMORY " 1962 YAMAUCHI, H ET AL " THE KT PILOT COMPUTER A MICROPROGRAMMED COMPUTER WITH A PHOTOTRAN

" 1968 33

CONTI,G; GIBSON, D & PITKOWSKY, S " STRUCTURAL ASPECTS OF THE SYSTEM 360; MODEL 85 " 1968 23

VAN DER POEL, W " MICRO-PROGRAMMING AND TRICKOLOGY " 1962 75

SEMPENE, H & PORETE, R " A STORED LOGIC COMPUTER " 1961 67

DREYER, K " PRINCIPLES OF A TWO-LEVEL MEMORY COMPUTER " 1968 25

BRILEY, D " PRIIPROGRAMMING " 1965 16

HAWRYLSZIKIEWICZ, A " MICROPROGRAMMED CONTROL IN PROBLEM-ORIENTED LANGUAGES " 1967 35

MELBOURNE, A & PUGMIRE, A " A SMALL COMPUTER FOR THE DIRECT PROCESSING OF FORTRAN STATEMENTS " 1965 54

IBM " DATA PROCESSING: MICROPROGRAMMING PI COURSE " 39

BECKL, K & KEELER, F " THE L-8601 DATA PROCESSOR " 1964 7

IN IMPLEMENTING AN ASSOCIATIVE PROCESSOR FOR A TIME-SHARED PRINCESSOR " 1968 HJERNES, G ET AL " CONTROL STORAGE USE 37
IN IMPLEMENTING AN ASSOCIATIVE PROCESSOR FOR A TIME-SHARED
PROCESSOR " 1968
HELLEMANN, L ET AL * CONTROL STORAGE USE
ET AL " CONTROL STORAGE USE IN IMPLEMENTING AN ASSOCIATIVE
PROCESSOR FOR A TIME-SHARED PROCESSOR " 1968
HORENES, G
LIVENCE, H " A SYSTEM/360
GRASSELLI, A " THE DESIGN OF
HILL " A SERIES/360 LOGIC PROGRAMMING AND APPLICATIONS" 1964
IVERSON, K " A
GREEN, J " MICROPROGRAMMING, EMULATORS AND
BORSCHY, W; VASILEVSKIY, P & KHMOROVIC, " A
LAWSON, H " PROGRAMMING-LANGUAGE-ORIENTED INSTRUCTION STREAMS" 1968
MELBOURNE, A & PUGMIRE, J " A SMALL COMPUTER FOR THE DIRECT PROCESSING OF DIS
RTRAN STATEMENTS " 1965
REBOULOT, M " EXTRACTEUR MICROPROGRAMME D' INFORMATION
ION CONCEPT " 1969
DOLBRECHT, J " MIUAS: MICROPROGRAMMING DESIGN AID SYSTEM FOR
BENJAMIN, R " THE SPECTRA 70/45 EMULATOR FOR THE
BAILEY, S " FASTER COMPUTER CONTROL WITH A
KLEIN, S & SCHWARTZ, S " MODEL 4200 - B200
LIVENCE, H " A SYSTEM/360 PROFILE —
UPPER, A " FOURTH-GENERATION SOFTWARE: THE
HECKELMAN, R " SELF-CHECKING
ROSEN, S " HARDWARE DESIGN
MILLER, E " 1968
ALLEN-BABCOCK COMPUTER INC " INFORMATION OUTLINE:
DECISION SYSTEMS, INC " INTERIM STATUS REPORT NO. 1 " 1968
ROSEN, S " HARDWARE DESIGN REFLECTING SOFTWARE
FLYNN, M & MACLAREN, M " MICROPROGRAMMING
E COMPUTER " 1967
COMPUTER WITH MICROPROGRAM CONTROL " 1963
WITH MICROPROGRAM CONTROL " 1963
RUSE, G; BASHKOWITZ, T; FELDMAN, M; SIEGEL, M & WOLLAN, S " SYSTEM ARCHITECTURE FOR LARGE
SM SYSTEM/360 MODEL 30 " 1965
MICMAC, M; SIEGEL, M; WOLLAN, S & WOLFE, M " THE FIRST-GENERATION INTERFACES
KLEIN, S & PETERSON, H " MICROPROGRAM CONTROL FOR THE EXPERIMENTAL
HECKELMAN, P " SELF-CHECKING
ICROMACHINE UNIVERSAL STRUCTURE STUDY ORIENTED TO COMPUTERS
TS " 1965
MELBOURNE, A & PUGMIRE, J " A
MECLURE, R & DAVIS, R " A
ROSEN, S " HARDWARE DESIGN REFLECTING
OPLER, A " NEW DIRECTIONS IN
HARDWARE AND SOFTWARE INTERFACES; A MACHINE ORGANIZATION SOLUTION, THE
INGALL, S " LOGICAL DESIGN OF A MICROPROGRAMMED
CAMPBELL, C & NEILSON, D " MICROPROGRAMMING THE SPECTRA 70/35 " 1966
A VARIOUS STRUCTURE DIGITAL COMPUTERS
<table>
<thead>
<tr>
<th>TEXT</th>
<th>KEYWORD AND TEXT</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLANKENHAGEN, J &quot;LOGICALLY MICRO-PROGRAMMED COMPUTERS&quot;</td>
<td>&quot;1960&quot;</td>
<td>11</td>
</tr>
<tr>
<td>OPLER, A &quot;NEW DIRECTIONS IN SOFTWARE CONTROL&quot; &quot;MICROPROGRAMMING&quot;</td>
<td>&quot;1960&quot;</td>
<td>59</td>
</tr>
<tr>
<td>CONROY, E &quot;MICROPROGRAMMING&quot;</td>
<td>&quot;1961&quot;</td>
<td>21</td>
</tr>
<tr>
<td>CONROY, E &amp; MEADE, R &quot;A MICROINSTRUCTION SYSTEM&quot;</td>
<td>&quot;1961&quot;</td>
<td>22</td>
</tr>
<tr>
<td>SEMARNE, H &amp; PORTER, R &quot;A STORED LOGIC COMPUTER&quot;</td>
<td>&quot;1961&quot;</td>
<td>67</td>
</tr>
<tr>
<td>SEMARNE, H &amp; MCGEA, W &quot;STORED LOGIC COMPUTING&quot;</td>
<td>&quot;1961&quot;</td>
<td>66</td>
</tr>
<tr>
<td>IVERSON, K &quot;A PROGRAMMING LANGUAGE&quot;</td>
<td>&quot;1962&quot;</td>
<td>41</td>
</tr>
<tr>
<td>VAN DER POEL, W &quot;MICRO-PROGRAMMING AND TRICKOLOGY&quot;</td>
<td>&quot;1962&quot;</td>
<td>75</td>
</tr>
<tr>
<td>SIGN OF PROGRAM-MODIFIABLE MICRO-PROGRAMMED CONTROL UNITS</td>
<td>&quot;1962&quot;</td>
<td>30</td>
</tr>
<tr>
<td>G. TIMOFEEV, A &quot;MICROPROGRAM CONTROL FOR DIGITAL COMPUTERS&quot;</td>
<td>&quot;1962&quot;</td>
<td>26</td>
</tr>
<tr>
<td>ROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY</td>
<td>&quot;1962&quot;</td>
<td>33</td>
</tr>
<tr>
<td>ROPROGRAMMED COMPUTER WITH A PHOTOTRANSISTOR FIXED MEMORY</td>
<td>&quot;1962&quot;</td>
<td>33</td>
</tr>
<tr>
<td>A 75/35 &quot;1966&quot; &quot;A VARIABLE STRUCTURE DIGITAL COMPUTER&quot;</td>
<td>&quot;1962&quot;</td>
<td>19</td>
</tr>
<tr>
<td>CAMPBELL, C &amp; NEILSON, D &quot;MICROPROGRAMMING THE SPECTRUM&quot;</td>
<td>&quot;1962&quot;</td>
<td>33</td>
</tr>
<tr>
<td>GEPACE, G &quot;MICROPROGRAMMED CONTROL FOR COMPUTING&quot;</td>
<td>&quot;1962&quot;</td>
<td>28</td>
</tr>
<tr>
<td>SCHAEPP, H &quot;MICROPROGRAMMED COMPUTER CONTROL&quot;</td>
<td>&quot;1963&quot;</td>
<td>65</td>
</tr>
<tr>
<td>RANDOL, Y &quot;EVOLUTION OF MICROPROGRAMMING CONCEPTS&quot;</td>
<td>&quot;1963&quot;</td>
<td>34</td>
</tr>
<tr>
<td>CALCULATION OF THE PB 440 MICROPROGRAMMABLE COMPUTER</td>
<td>&quot;1963&quot;</td>
<td>14</td>
</tr>
<tr>
<td>CONOMICAL MULTIPROGRAM COMPUTER WITH MICROPROGRAM CONTROL</td>
<td>&quot;1963&quot;</td>
<td>7</td>
</tr>
<tr>
<td>CONOMICAL MULTIPROGRAM COMPUTER WITH MICROPROGRAM CONTROL</td>
<td>&quot;1963&quot;</td>
<td>12</td>
</tr>
<tr>
<td>HILL, R &quot;STORED LOGIC PROGRAMMING AND APPLICATIONS&quot;</td>
<td>&quot;1964&quot;</td>
<td>38</td>
</tr>
<tr>
<td>BOUTVEL, E &quot;THE PB 440 COMPUTER&quot;</td>
<td>&quot;1964&quot;</td>
<td>13</td>
</tr>
<tr>
<td>BECK, L &amp; KEELER, F &quot;THE C-8401 DATA PROCESSOR&quot;</td>
<td>&quot;1964&quot;</td>
<td>50</td>
</tr>
<tr>
<td>MCGE, W &quot;THE TRW-133 COMPUTER&quot;</td>
<td>&quot;1964&quot;</td>
<td>3</td>
</tr>
<tr>
<td>AMDahl, L &quot;MICROPROGRAMMING AND STORED LOGIC&quot;</td>
<td>&quot;1964&quot;</td>
<td>71</td>
</tr>
<tr>
<td>CHARACTER COMPUTER FOR HIGH LEVEL LANGUAGE INTERPRETATION</td>
<td>&quot;1964&quot;</td>
<td>53</td>
</tr>
<tr>
<td>STRUCTURE OF SYSTEM/360; PART II - SYSTEM IMPLEMENTATIONS</td>
<td>&quot;1964&quot;</td>
<td>70</td>
</tr>
<tr>
<td>BRILEY, B &quot;PRIOR PROGRAMMING&quot;</td>
<td>&quot;1965&quot;</td>
<td>15</td>
</tr>
<tr>
<td>BENJAMIN, R &quot;THE SPECTRA 70/45 EMULATOR FOR THE RCA 301&quot;</td>
<td>&quot;1965&quot;</td>
<td>46</td>
</tr>
<tr>
<td>LIVERANCE, H &quot;A SYSTEM/360 PROFILE -- READ-ONLY STORAGE&quot;</td>
<td>&quot;1965&quot;</td>
<td>45</td>
</tr>
<tr>
<td>LEYF, L &quot;STATE OF THE ART MICROPROGRAMMING&quot;</td>
<td>&quot;1965&quot;</td>
<td>9</td>
</tr>
<tr>
<td>SDN, H &quot;MICROPROGRAMMING FOR DATA ACQUISITION AND CONTROL&quot;</td>
<td>&quot;1965&quot;</td>
<td>52</td>
</tr>
<tr>
<td>ON, H &quot;MICROPROGRAM CONTROL FOR THE EXPERIMENTAL SCIENCES&quot;</td>
<td>&quot;1965&quot;</td>
<td>52</td>
</tr>
<tr>
<td>W. CRAFORD &amp; NOWELL, G &quot;THE CONTROLS AUTOMATION SYSTEM&quot;</td>
<td>&quot;1965&quot;</td>
<td>17</td>
</tr>
<tr>
<td>COMPUTER ARCHITECTURE AND ITS DIRECT PROCESSING OF FORTRAN STATEMENTS</td>
<td>&quot;1965&quot;</td>
<td>56</td>
</tr>
<tr>
<td>1401 COMPATIBILITY FEATURE ON THE IBM SYSTEM/360 MODEL 30&quot;</td>
<td>&quot;1965&quot;</td>
<td>49</td>
</tr>
<tr>
<td>HECKELMAN, R &quot;SELF-CHECKING REDUNDANT MICROPROGRAMS&quot;</td>
<td>&quot;1966&quot;</td>
<td>36</td>
</tr>
<tr>
<td>OPLER, A &quot;NEW DIRECTIONS IN SOFTWARE 1960-1966&quot;</td>
<td>&quot;1966&quot;</td>
<td>59</td>
</tr>
<tr>
<td>J &quot;MICROPROGRAMMING, EMULATORS AND PROGRAMMING LANGUAGES&quot;</td>
<td>&quot;1966&quot;</td>
<td>31</td>
</tr>
<tr>
<td>I AL DESIGN OF A MICROPROGRAMMED SPECIAL-PURPOSE COMPUTER</td>
<td>&quot;1966&quot;</td>
<td>40</td>
</tr>
<tr>
<td>P E KHOIYAKOV, A &quot;A PROGRAMMING MICRO-LANGUAGE FOR URAL-4&quot;</td>
<td>&quot;1966&quot;</td>
<td>12</td>
</tr>
<tr>
<td>MPBELL, G &amp; NEILSON, D &quot;MICROPROGRAMMING THE SPECTRA 70/35&quot;</td>
<td>&quot;1966&quot;</td>
<td>19</td>
</tr>
<tr>
<td>N A VARIABLE STRUCTURE DIGITAL COMPUTER &quot;1966&quot;</td>
<td>&quot;1966&quot;</td>
<td>69</td>
</tr>
<tr>
<td>ORGANIZATION SOLUTION, THE VARIABLE INSTRUCTION COMPUTER</td>
<td>&quot;1966&quot;</td>
<td>69</td>
</tr>
<tr>
<td>OPLER, A &quot;FOURTH-GENERATION SOFTWARE: THE REALIGNMENT&quot;</td>
<td>&quot;1967&quot;</td>
<td>58</td>
</tr>
<tr>
<td>BREUER, N &quot;ADAPTIVE COMPUTERS&quot;</td>
<td>&quot;1967&quot;</td>
<td>13</td>
</tr>
<tr>
<td>TUCKER, S &quot;MICROPROGRAM CONTROL FOR SYSTEM/360&quot;</td>
<td>&quot;1967&quot;</td>
<td>73</td>
</tr>
<tr>
<td>SLOBDYANYUK, J &quot;A METHOD OF MINIMIZING MICROPROGRAMS&quot;</td>
<td>&quot;1967&quot;</td>
<td>68</td>
</tr>
<tr>
<td>FLYNN, W &amp; MAJACRA, N &quot;MICROPROGRAMMING VISITED&quot;</td>
<td>&quot;1967&quot;</td>
<td>27</td>
</tr>
<tr>
<td>AILEY, S &quot;FASTER COMPUTER CONTROL WITH A READ-ONLY MEMORY&quot;</td>
<td>&quot;1967&quot;</td>
<td>21</td>
</tr>
<tr>
<td>W &amp; VANDLING, G &quot;SYSTEM IMPLICATIONS OF MICROPROGRAMS&quot;</td>
<td>&quot;1967&quot;</td>
<td>74</td>
</tr>
<tr>
<td>S J &quot;MODULAR COMPUTER DESIGN WITH PICROPROGRAMMED CONTROL&quot;</td>
<td>&quot;1967&quot;</td>
<td>56</td>
</tr>
<tr>
<td>RELIABILITY ASPECTS OF THE VARIABLE INSTRUCTION COMPUTER</td>
<td>&quot;1967&quot;</td>
<td>56</td>
</tr>
</tbody>
</table>
### ANNOTATED MICROPROGRAMMING BIBLIOGRAPHY

This bibliography provides an indexed, annotated list of publications related to the field of microprogramming, from the introduction of the term in 1951 to the present time.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
</tr>
<tr>
<td>MICROPROGRAMMING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICROPROGRAMMING BIBLIOGRAPHY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANNOTATED MICROPROGRAMMING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
REPLY TO
ATTN OF:  ESGEI/5322

SUBJECT:  Review of Technical Document

TO:  ESTI

Identification:  ESD-TR-70-204

ESAI has no objection to release of the above-cited document to the Clearinghouse for Scientific and Technical Information.

J. O'Brien
Chief, Public Information Division
Information Office