AD-A091 242 HAWAII UNIV HONOLULU DEPT OF ELECTRICAL ENGINEERING F/G 9/2
COMPUTER ORGANIZATION AND INFORMATION HIDING.(U)
JUL 80 R CHATTERGY DASG60-79-C-0118
UNCLASSIFIED NL

END
DATE FILMED
12-80
DTIC

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

REPORT NUMBER: Interim Report No. 3	2. GOVT ACCESSION NO.: AD-A091242
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Computer Organization and Information Hiding	5. TYPE OF REPORT & PERIOD COVERED: Interim, Third Quarter
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): R. Chatterjy (10) Rahul Chatterjy	8. CONTRACT OR GRANT NUMBER(s): (15) DASG60-79-C-0118
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Dept. of Electrical Engineering, University of Hawaii, 2540 Dole St. Hon., HI 96822	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (11) 15 Jul 80
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Technology Center, U.S. Army Ballistic Missile Defense, P.O. Box 1500 Huntsville, Alabama 35807	12. REPORT DATE: 7/15/80 (12) 18
	13. NUMBER OF PAGES: 16
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, 1030 E. Green St., Pasadena, Calif. 91106	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DTIC ELECTE NOV 4 1980 S D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

The views, opinions, and/or findings contained in this report are those of the author and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other official documentation.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer organization, Information hiding, Coupling, Modules, Architecture.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A new internal organization of a computer is proposed by applying the principle of information hiding to the structure of a computer. The new organization is inspected in the light of the requirements set forth on computer organization by Functional Programming systems. The current status of the implementation of such a computer and the future directions of research are mentioned.

DD FORM 1 JAN 73 1473

400 641

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

COMPUTER ORGANIZATION AND INFORMATION HIDING[†]

RAHUL CHATTERGY[††]

Abstract--A new internal organization of a computer is proposed by applying the principle of information hiding to the structure of a computer. The new organization is inspected in the light of the requirements set forth on computer organization by Functional Programming systems. The currect status of the implementation of such a computer and the future directions of research are mentioned.

Index Terms - Computer organization, Information hiding, Coupling, Modules, Architecture.

[†]Sponsored by The Ballistic Missile Defense Advanced Technology Center, Huntsville, Alabama, under Contract No. DASG60-79-C-0118.

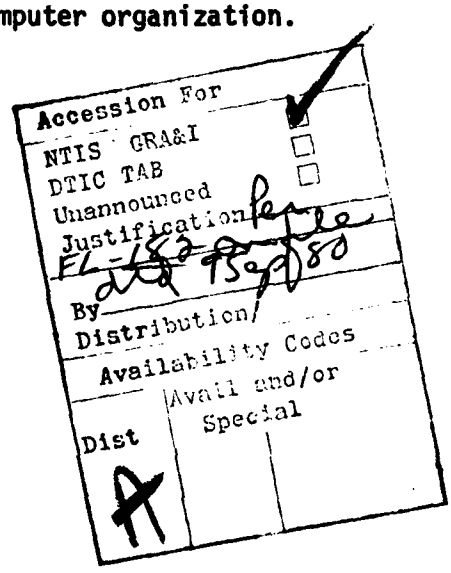
[††]Department of Electrical Engineering, University of Hawaii, 2540 Dole Street, Honolulu, HI 96822.

I. INTRODUCTION

The internal organization of a computer has remained unchanged since the first stored-program digital computers EDVAC and EDSAC were designed. With the passage of time the complexity, the size and the response-time requirements of the programs have enormously increased. The stringent demands placed on the digital computers have uncovered many of their organizational limitations on performance. Various modifications have been developed to alleviate these restrictions on performance with attendant problems of their own.

Our purpose is to uncover the basic principle (if any) behind the organization of the existing computers and then to search for new principles of organization of complex systems. We contend that some of the limitations of the current organization are inherent in the principle used to create such an organization. Only alternate principles of organization can remove these limitations and produce radically different internal organizations of computers.

We believe that Parnas' principle of information hiding [1], [2] is such an alternate principle. This principle originated in software engineering, but portends new structures for computer organization.



II. THE CONVENTIONAL ORGANIZATION

Let us consider some of the current computational demands on the existing computer systems. One such demand is the creation of a computational environment for real-time computing. The following set of requirements for real-time computing are obtained from [3]: (i) large databases created in real-time, (ii) routines of enormous size, (iii) totally automated system requiring a high degree of reliability, and (iv) real-time response in a rapidly changing environment.

Distributed processing [5] is examined in [4] as a possible approach to real-time data processing. The basic idea consists of distributing processing power and making it available at the point where data are stored. The motivating factors for a distributed processing system are the ease of system modification, reconfiguration, graceful degradation, and an increase in throughput with a corresponding reduction in response-time.

Until now, to our knowledge all processing systems, no matter how novel, have been based on the conventional Mauchly-Eckert-von Neumann organization of a computer. This organization is based on controlled exchanges among a storage unit (STU), a processing unit (PRU), and an input/output unit (IOU). This organization is further characterized by strong couplings [6] via large volumes of exchanged information among these units.

The network approach to distributed processing employs computers with conventional organization as host processors. Multiprocessor systems multiply the number of the STUs and the PRUs connected through

a switching network or a shared bus structure with a tendency to saturate quickly under the increased volume of transfers.

The conventional organization has certain limitations which can not be eliminated by cleverly designed interconnection networks. Some of these limitations and attempts to alleviate their effects are discussed in the next section.

III. LIMITATIONS OF THE CONVENTIONAL ORGANIZATION

The following is a partial list of the limitations of the conventional organization: (i) storage bottleneck, (ii) difficulty in programming, (iii) limited flexibility for multiprocessing, and (iv) long processing time. These limitations and their implications have been discussed by various people in the past [7, 8, 9].

Let us mention just one important implication of the conventional organization, pointed out by Backus [9]. This organization has affected the structure of all programming languages developed in the past. A conventional programming language is basically a high-level, complex version of the Mauchly-Eckert-von Neumann computer. From this point of view one can say that, reducing the semantic gap between a conventional programming system and a computer may make programming somewhat easier, but it will not create a radically new programming system.

The storage bottleneck mentioned earlier is a major limitation of the conventional organization. It is in part responsible for the limited flexibility for multiprocessing and the long processing time.

One approach to reduce storage access-delay uses interleaved addresses and a recirculating buffer to store addresses [10]. The resulting problem of the alteration in the sequence in which incoming addresses are accepted by the storage system is resolved by introducing read/write distributors.

The increased storage bandwidth so obtained is utilized by allowing look-ahead fetch of the instructions from the storage unit. The look-ahead fetch technique is based on the empirical principle of locality exhibited in most instruction streams. The presence of branch

instructions gives rise to the problem of buffer management and a scheme for that is described in [11].

To take advantage of a rapid stream of instructions, one either uses a pipelined processor [12] or a concurrent-execution processor [10]. Each processor has its own set of control problems and are controlled in complex ways (see [10, 11, 12]).

Reduction of processing time has long been a major objective for the design of scientific processors. Cray-1 [13] is a current example of such a processor which uses a large local store in the processor, and a technique known as chaining for reducing processing time.

We hope that this short discussion demonstrates that the conventional organization of a computer has serious limitations. Complex control schemes have been invented to moderate the effects of some of the limitations. Some of the implications of these limitations are very disturbing indeed for they influence our way of thinking about programs and programming [9].

IV. FUNCTIONAL DECOMPOSITION

Let us now see if we can find any principle behind the organization of the conventional computer. If no such principle exists, then we are doomed to inventing ad hoc methods for solving one problem created by the solution of another. However, if a principle can be identified, we can look for alternate principles and hope to find a better one.

A computer is a complex system and, therefore, it is probably organized by using some principle used to organize other complex systems. One such principle, used from time immemorial, is the principle of decomposition. A complex system is decomposed into modules where each module represents a simpler subsystem. The process of decomposition may be hierarchical or not [14] and may consist of several levels.

But how do we define the boundary of each module? An intuitive and very commonly used approach is to define a module by its function. We call this mode of decomposition, functional decomposition, and derive from it the principle of functional decomposition.

Let us now apply this principle of functional decomposition to the organization of a computer. The overall function of a computer can be decomposed into a STORE function and a PROCESS function if we ignore input/output for the moment. The STORE function is implemented in a hardware STU module. This module needs only a medium of storage and some hardware for storing and retrieving information in selected locations.

The PROCESS function is implemented in a hardware PRU module. The basic functions of this module is to differentiate among the instructions and the operands, fetch the instructions in a prescribed sequence,

fetch the operands as necessary, and execute the instructions. As a consequence, a PRU has hardware for instruction flow control, address computation, local storage, local storage control, a arithmetic/logic unit (ALU), and ALU operations control. Furthermore it has control circuits for controlling a IOU which operates asynchronously from the PRU and the STU.

Programmers often use the principle of functional decomposition to modularize large programs. A measure of success with such an approach is the number of modules (often subroutines) that can be shared to perform different functions. This is no different from the decomposition described in the previous paragraphs. The control circuits for address computation, instruction flow control and IOU control in the PRU are similar to the shared subroutines. They are shared among different programs in the STU to carry out different overall functions.

The reason for the storage bottleneck is now obvious. Most of the control functions are concentrated in the PRU, and the objects being controlled are mostly in the STU. Most of the traffic between the STU and the PRU consists not of the objects being operated upon, but information to be processed by the control functions to identify these objects [9].

V. INFORMATION HIDING

In the our opinion, a truly new principle of decompostion was first enunciated by David Parnas in [1]. The modules obtained by using this principle do not necessarily correspond to the functions carried out in a computational system. Each module organizes and manipulates data structures which it hides from the other modules and hence, the name principle of information hiding. From a broader perspective, each module hides certain design decisions from other modules.

A specific guideline for system decomposition, that originates from the principle of information hiding, is to make a data structure, its internal linkings, accessing, and modifying procedures, part of a single module. Such a guideline can be used for the decomposition of any complex system, not necessarily just software. We shall use this guideline to decompose the hardware system of a computer.

A basic information structure in a computational environment is a program which consists of a set of instructions and a set of operands. Hence, we create an instruction module which consists of a set of instructions along with the procedures for instruction flow control and address computation. Similarly we create an operand module which stores operands and incorporates operand accessing procedures. The processor module consists of the arithmetic-logic unit, local storage, and procedures for managing these resources.

In this decomposition of a computing system, the processor module receives instructions and operands and returns results and/or the status of an operation. It does not receive or process addresses. Addresses

are retained and processed only by those modules that need to process them.

Information must still be exchanged among the instruction module, the operand module and the processor module. However, they exchange only such information as is essential for the proper functioning of the overall system. Some addresses must be exchanged between the instruction module and the operand module. The number of bits transferred can be minimized by such conventional techniques as relative addressing. External information may need to be transferred to the processor module for the management of local storage. The amount of such information can be minimized if the local store is structured as a stack. Certain synchronization information must also be exchanged, but this is also true in a conventional organization.

The principle of information hiding generates modules which hide information, and tends to minimize information transfers among modules. Hence, it is perhaps not inappropriate to call these modules, latent-information modules and the resulting organization, minimum information-transfer (MIT) organization. Admittedly the implied minimization is not quantitative in nature. However, the benefits to be derived at this point from a purely ritualistic formalization and quantization is not clear. Hence, let us consider some of the implications of the minimum information-transfer organization.

IV. MINIMUM INFORMATION-TRANSFER ORGANIZATION

The MIT organization removes a bottleneck between the STU module and the PRU module which Backus has called the "von Neumann bottleneck" [9]. This is done by a redistribution of the processing power from one module to another, according to the principle of information hiding. A somewhat similar organization can be found in the Fairchild F8 microprocessor system.

Implimentation of latent-information modules by LSI circuit technology can alleviate the constraint of pin-limitation. For example, one version of a conventional microprocessor (MC68000) uses a 64 pin package, out of which, 32 pins are used for address transfers. With a MIT organization based on latent-information modules, such pins on the processor module can be made available for other uses.

Two of the most important uses of these pins are, (i) direct interaction with a stream of input/output data, and (ii) direct processor to processor communication. In the first case, an instruction module along with a processor module connected to suitable transducers can create a real-time processing system. Such a processing system may be useful in guidance and weapons systems where long-term storage of input data may be pointless. The second case opens up a new way of interconnecting processors that does not use shared memory accessed via an asynchronous, shared bus. Processors may be pipelined or interconnected according to other schemes for special purpose computing systems.

A MIT organization is also ideally suited for implementation by means of VLSI circuit technology. The chip area is better used since the

number of lines interconnecting latent-information modules are kept at a minimum by module design.

Certain other implications of the MIT organization have come to light during a recent experiment described in the next section.

VII. CURRENT STATUS OF RESEARCH

As an experiment in MIT organization, we have simulated a small uniprogrammed stack computer designed on the basis of the principle of information hiding. In programming this simulated machine we realized that the instruction module can be further decomposed into two submodules.

The instruction-stream module stores packets of instructions where, such packet has a definite beginning and an end. Instructions within a packet are executed sequentially. The purpose of the instruction-stream module is to supply a stream of instructions upon receiving a packet identification.

The flow-control module directs control through these packets. It stores machine language versions of higher level constructs such as, WHILE statements. The procedures in this module interpret these constructs and direct the flow out of the instruction-stream module.

The results of this simulation study will be discussed in the future. The discovery of the usefulness of the instruction-stream module, and the flow-control module leads us to believe that the natural mode of programming on a MIT organization is not the one-word at a time approach of von Neumann languages [9]. Each packet may change the contents of the store in a major way. Once such a packet is transmitted to a processor module, it is not interrrupted until the entire packet is done. Interrupts are channeled to the flow-control module. To summarize, some version of the MIT organization may be better suited to support a functional programming system [9] than a conventional organization.

Recently we have come to know [15] that Dougherty at the Franklin Research Center has implemented hardware systems that can be reconfigured into a MIT organization. Cooperative investigation of a MIT organization on this hardware system is being planned.

VIII. FUTURE DIRECTIONS

In essence, this paper presents a proposal in the same vein as in [16], for the organization of a new computing system. As yet, the design cannot be supported by empirical data and it will take time to gather such data. However, the organization is based on a radical application of a software design principle to hardware design. It appears to have far reaching implications for the LSI and VLSI circuit implementations of computing systems and also on their programming methods.

The general objective of this research effort is to arrive at a design technique that can generate a family of computers of varying processing power. Obviously this cannot be done by taking individual systems and reorganizing them according to the principle of information hiding.

A better approach is to design a family of operating systems [17] and to allow each member of the family to dictate the organization of its hardware. In [17], Parnas has described the virtual memory module of such a family. We are attempting to design the I/O module of such a family based on the concept of interrupt hiding [15, 18].

Finally, let us note that a MIT organization is not a virtual machine. To obtain a virtual machine we program a given hardware organization. A MIT organization is obtained by using a software design principle to dictate the organization of the underlying hardware.

REFERENCES

1. Parnas, D.L., "On the Criteria to be Used in Decomposing Systems into Modules," CACM, v. 5, no. 12, December 1972, pp. 1053-1058.

2. Parnas, D.L., "Information Distribution Aspects of Design Methodology," Proc. IFIP Congress, North-Holland, 1972, pp. 339-344.

3. Vick, C.R., "Requirements for a Real-Time Data Processing System," Proc. 11th Hawaii Intl. Conf. on System Science, January 1977.

4. Vick, C.R., J.R. Scalf and W.C. McDonald, "Distributed Data Processing for Real-Time Applications," Proc. Texas. Computer Conf., Austin, Texas, November 1977.

5. Rao, C.R. and C.V. Ramamoorthy, "The Design Issues in Distributed Computer Systems," Infotech, London, pp. 377-399.

6. Chattergy, R., "On the Concept of Coupling," Interim Report No. 2, BMDATC, May 1980.

7. Heath, F.G., "Entering the Non-von-Neumann Era," IEEE Computer and Digital Techniques," v. 2, no. 2, April 1979, pp. 57-58.

8. Thompson, T.R., Computers and Their Future, Speeches given at The World Computer Pioneer Conference, Llandudno, July 1970, Published by Richard Williams and Partners, G.P.O. Box 8, Llandudno, Wales, pp. 8/1-8/27.

9. Backus, J., "Can Programming be Liberated from the von Neumann Style?...," CACM, v. 21, no. 8, August 1978, pp. 613-640.

10. Thornton, J.E., "Design of a Computer, The Control Data 6600," Scott, Foresman, 1970.

11. Ibbett, R.N., "The MU5 Instruction Pipeline," Computer Journal, v. 15, 1972, pp. 43-50.

12. Ibbett, R.N. and P.C. Capon, "The Development of the MU5 Computer System," CACM, v. 21, no. 1, January 1978, pp. 13-24.

13. Russell, R.M., "The Cray-1 Computer System," CACM, v. 21, no. 1, January 1978, pp. 63-72,

14. Parnas, D., "On a 'Buzzword': Hierarchical Structure," Proc. IFIP Congress, North-Holland, 1974, pp. 336-339.

15. Dougherty, E.J., Private Communication, Franklin Research Center, Philadelphia.

16. Kilburn, T. et al., "A System Design Proposal," Proc. IFIP Congress, Edinburgh, August 1968.

17. Parnas, D.L. et al., "Design and Specification of the Minimal Subset of an Operating System Family," Trans. IEEE Software Engineering, v. SE-2, no. 4, December 1976, pp. 301-307.

18. Wirth, N., "On Multiprogramming, Machine Coding, and Computer Organization," CACM, v. 12, no. 9, September 1969, pp. 489-498, (Corrections, v. 13, no. 4, April 1970, p. 266).