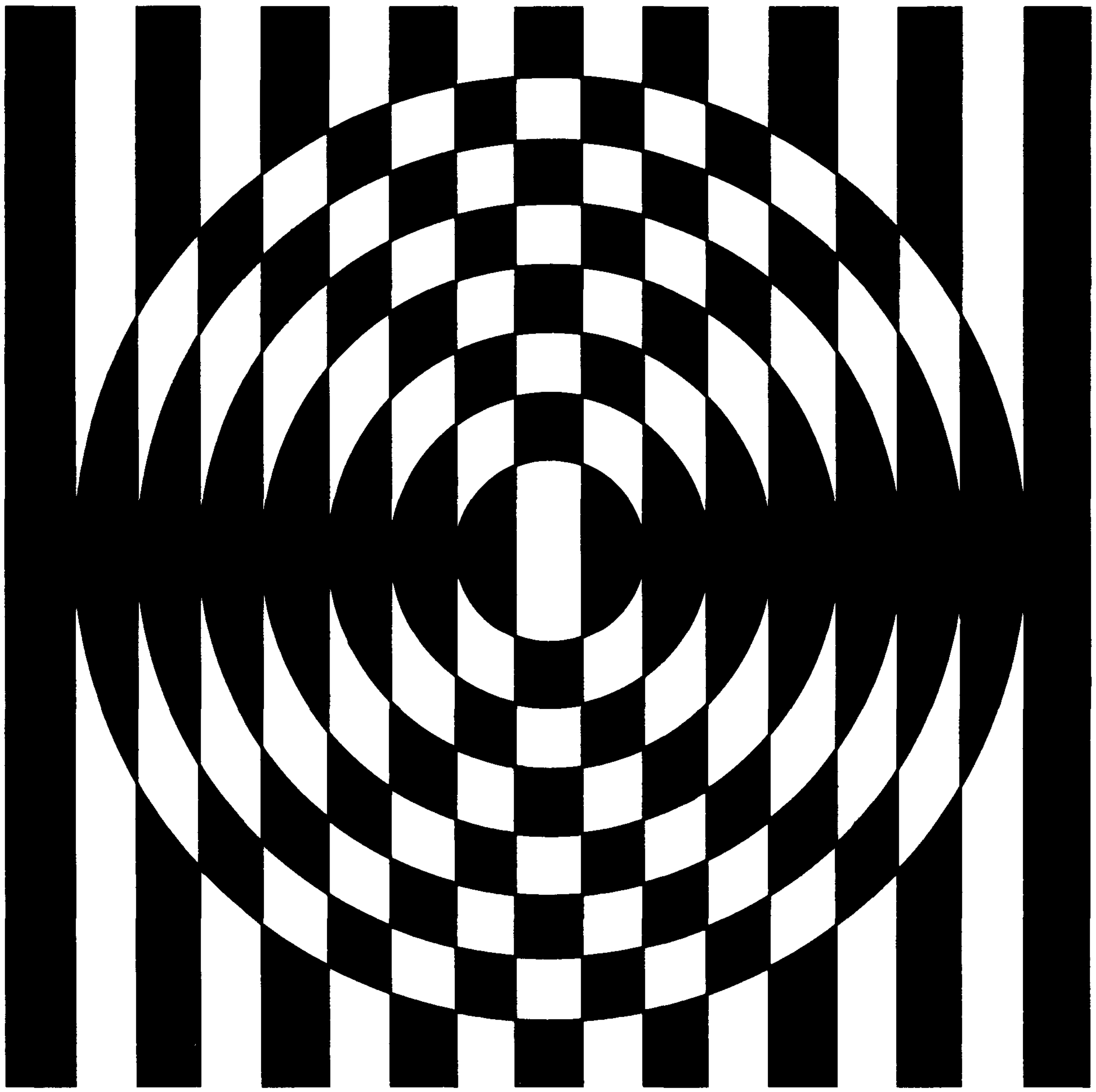
①

JB.

An Undergraduate Electrical Engineering Course On Computer Organization

October 1968

Cosine Committee Commission on Engineering Education



AN UNDERGRADUATE ELECTRICAL ENGINEERING COURSE ON COMPUTER ORGANIZATION

An Interim Report of the

COSINE COMMITTEE

of the

COMMISSION ON ENGINEERING EDUCATION
2101 Constitution Avenue
Washington, D.C. 20418

October, 1968

Task Force on the Computer Organization Course
COSINE Committee
Commission on Engineering Education

C. Gordon Bell, Carnegie-Mellon University
Yaohan Chu, University of Maryland
C. L. Coates, University of Texas
Wayne Lichtenberger, University of California, Berkeley
Fred Luconi, Massachusetts Institute of Technology
William Viavant, University of Utah
E. J. McCluskey, Stanford University, Chairman

This report has been supported in part by the National Science Foundation under Grant GY-2108.

An Undergraduate Electrical Engineering Course on

COMPUTER ORGANIZATION:

Report by a COSINE Task Force

I. INTRODUCTION

Although new departments are being started to educate undergraduates specifically in the area of Computer Science, there is an ever increasing need for electrical engineers whose undergraduate program provided a familiarity with digital system design. This need arises because of the increasing demand for special purpose digital data processing and control systems as well as for general purpose digital computers. To meet the demand it will be necessary to offer the undergraduate electrical engineering student the opportunity of an option or elective program in the appropriate subject matter. Such a program must include both the hardware and software aspects of digital systems that are essential to the design function.

One course that is fundamental to such a program is concerned with the elements of computer organization. Such a course should be offered by the Electrical Engineering Department and should correlate the design and organizational aspects of the subject.

The content of such a course is the topic of this report which was written as a result of a two day meeting held at Stanford University on July 18, 19, 1968. On these days a group of seven educators experienced in Computer Science and Electrical Engineering held discussions concerning a Computer Organization course. In addition a dinner meeting was held with local representatives of the computer industry and universities. The results of these discussions are presented on the following pages.

The group of educators responsible for this report was acting under the sponsorship of the COSINE Committee of the Commission on Engineering Education and was constituted as the Task Force on the Computer Organization Course. Parallel activities are being carried on by the Task Force on the First Course and the Task Force on Digital Subsystems.

The purpose of this report is twofold: First, it is hoped that it will be of considerable direct help to university faculty members wishing to teach a course such as the one described. To facilitate this use, detailed specific references to available books and articles are given for each of the topics recommended. Second, it is expected that the discussions presented here will stimulate the preparation of textbooks written explicitly for courses such as the one covered here.

II. BACKGROUND

The course described here is assumed to be an upper-class undergraduate *elective* course in Electrical Engineering for students with a major interest in digital systems. Before considering the course content, a discussion was held concerning the prerequisites or background knowledge the students taking such a course could reasonably be assumed to have.

There was quick agreement that all students should have had an "Introduction to Computing" course in which they learned to write programs in an algorithmic language. A preference was expressed for their learning ALGOL or PL/1, but because of the practicalities involved it was recognized that in many cases the language covered in such a course would be FORTRAN.

It was further agreed that it is desirable that a course in assembly language programming precede or be taken concurrently with the computer organization course. The group, recognizing that this might not be possible in all cases, therefore arranged the computer organization course so that assembly language programming was not an essential prerequisite.

Since one objective of the computer organization course is the design function, it is necessary that the course either be preceded by at least one course in logic design or that logic design topics be presented in the core electrical engineering courses. Table I lists the topics that should be included and that are considered prerequisite to the computer organization course. An adequate *level* for these topics is approximately that of either Bartee's book* or Maley and Heilweil's book**. It was also agreed that additional courses in logic design and switching theory should be available as electives.

TABLE I: Prerequisite Logic Design Topics

1. Number Systems and Codes
2. Binary Arithmetic
3. Boolean Algebra
4. Logic Elements – AND, OR, FLIP-FLOP (electrical circuit diagrams as well as logical performance)
5. Combinational Circuits (including Karnaugh Maps)
6. Shift Registers, Counters, Adders, Decoders
7. Registers and Information Transfer

RECOMMENDATION

The logic design topics listed in Table I are prerequisites to the computer organization course and, moreover, are considered sufficiently important to be required of all students who seek the BSEE degree. In addition, elective courses covering logic design and switching theory in more depth are desirable.

*Bartee, T. C., *Digital Computer Fundamentals,* 2nd Editic McGraw-Hill Book Co., New York, 1966.

**Maley, G. A., Heilweil, M., *Introduction To Digital Computers,* Prentice-Hall, Englewood Cliffs, N. J., 1968.

III. SYLLABUS FOR COMPUTER ORGANIZATION COURSE

The following is a presentation of the syllabus for the Computer Organization course which was arrived at after considerable discussion. Each of the topics has specific references given to indicate more precisely the items to be covered and to provide text material.

SYLLABUS:

I. A SIMPLE STORED PROGRAM COMPUTER*

A. Stored-program concept

B. Configuration (registers, adders, switches, etc.)

C. Instruction and data formats

D. Instruction set
[1: Ch7], [13: Sec11.1-11.5], [18]

E. Programming and software
[14: Ch18]

II. DATA REPRESENTATION AND ALGORITHMS FOR OPERATING ON DATA

A. Data and common number systems (e.g., integers, floating point, character strings)

B. Operations on data

I. Algorithms for arithmetic and logical data

2. Expression of algorithms using programming languages and flow charts

3. Expression of algorithms in hardware

a. Conventional

b. Macro modules
[1: Sec 8.1.1-8.1.6], [2], [3]

III. COMPUTER UNITS

A. Processors

1. Instruction formats and repertoire
[1: Sec 8.2.4], [13: Sec 12.5]

2. Fixed-point arithmetic unit
[1: Sec 8.1.7], [13: Sec 12.1-12.2]

3. Floating-point arithmetic unit
[1: Sec 8.1.6], [13: Sec 12.4]

B. Control units
[13: Sec 12.6]

1. Timing and control signals
[1: Sec 8.2.1], [13: Sec 10.8]

2. Instruction decoding
[1: Sec 8.2.2]

3. Addressing and indexing
[1: Secs 8.2.3, 8.2.5]

4. Instruction and execution cycles
[13: Sec 11.4], [14: Ch 15]

5. Interrupt and priority
[14: Sec 17.8]

6. Microprogramming
[1: Sec 9.5], [13: Secs 12.7, 12.8]

C. Memory units
[1: Secs 8.3, 10.1, 10.2], [5], [14: Ch16], [15: Sec 5.3]

1. Coincident current magnetic core memory
[1: Sec 8.3.1], [16: Ch 24]

2. Linear selection magnetic core memory
[1: Sec 8.3.2], [16: Ch 24]

3. 2½ D magnetic core memory
[16: Ch 24], [17]

4. Drum and disk memories
[1: Sec 8.3.4], [14: Sec 17.7]

5. Associative memories
[1: Sec 8.3.5]

6. Memory control
[4: Sec 3.3-3.5, Sec 5.7-5.10]

7. Memory bus
[1: Sec 8.5], [4: Sec 5.1-5.3]

D. Channels and I/O units
[1: Sec 8.4], [4: Sec 9.7], [14: Ch 17]

IV. THE COMBINATION OF COMPUTER COMPONENTS TO FORM A STRUCTURE

A. Simplex or 1 processor with ability to carry out all tasks
[1: Sec 8.5]

B. The uni-arithmetic processor, multiple channel structure
[1: Sec 8.5], [4: Ch 8]

*The instructor may choose a commercially available computer and use the manual supplied by the manufacturer.

*C. Memory mapping, multiprogramming and resource allocation
[6]

*D. Multi-processor structures
[1: Sec 8.5]

*E. Time sharing
[7]

F. Reliability in a structure
[1: Sec 10.3], [4: Ch 10]

V. SELECTED COMPUTER EXAMPLES

A. Classical von Neumann machine
[8], [9]

B. B 5000 – A stack organized multi-processor system
[10]

*C. CDC 6600
[11]

*D. IBM 360
[12]

IV. PROJECT

In order to become familiar with computer organization, projects in simulating functional operations of a computer on an available computer are advisable whenever possible. Examples of such projects are: simulation of a fetch sequence with indexing and indirect addressing; simulation of an addition, a subtraction, a multiplication, or a division sequence; simulations of a main control sequence with interrupt and priority feature; simulation of a simple but non-trivial stored program computer.

To simulate a digital computer on another computer one has to select a simulation language and its associated simulator. Two possible choices are described below.

The Use of a General Purpose Programming Language

One choice is to select an available programming language whose compiler is available in the systems of the university computer installation. The most common such languages are FORTRAN, MAD, ALGOL, and PL/1. Because of availability of one or more of these languages in many universities, this is the most practical choice. These programming languages are designed for computational purposes. There will be some difficulties in describing the computer logic and function. An example of the difficulty is the description of parallel micro-operations which occur in a computer organization.

The Use of a Simulation Language

A computer simulation language is designed specially for describing the function and organization of a computer. As a result, most of the difficulties that are encounted in using an available programming language are removed. These languages differ from one another in the degree of closeness to the physical implementation and in the generality of describing computer function and organization. Many such languages have been reported in the literature, but most have no simulators.

*Optional, may be left for more advanced course

V. DISCUSSION

After taking this course a student should be able to identify the functional units of an information processing system. He should understand alternative designs for each functional unit and should know the basic economic and technical factors on which to base a choice of alternatives.

He should know the organization of several existing computers and should be able to read papers and manuals describing new ones. If given specifications for a simple computer, the student should be able to create a functional design and justify it technically. He should also be able to simulate his design with a program written in a procedural or simulation language.

He should appreciate the dependence of programming on computer organization, and consequently, the dependence of new organizations on programming needs. He should understand how the division between hardware and software depends on changes in circuit and component technology.

Most texts describe computer organization with a tendency to concentrate on a particular organization (usually a simple von Neumann machine) and show very few alternatives. To offer perspective, it is suggested that a number of examples of different types of machines be discussed explicitly in the latte part of the course (V, Selected Computer Examples). The students should be required to read the references, with the instructor filling in background and smoothing over differences in terminology. The references and what is intended to be conveyed in them are described below:

A. References [8] and [9] discuss the classical von Neumann machine. These references are not much different from the text, but the machine being discussed is complete and real.

B. The B 5000 is representative of a daring departure from the von Neumann-type machine. The principal feature of the machine is the use of the stack mechanism. Reference [10] describes design objectives and system organization in a very readable fashion.

C. The CDC 6000 series [11] machines display two interesting features which should be emphasized. The use of a set of small peripheral processors which control the flow of information between various components of the system is important. The method of addressing central registers in the central processor is also novel and fairly general.

D. System 360 [12], [4: Ch 9] represents the most widely used third generation system. Many schools may have such a system on campus; hence there may be particular motivation for coverage of this material.

REFERENCES

[1] Gschwind, H. W., *Design of Digital Computers,* Springer--Verlag, New York, 1967.

[2] Clark, W. A., "Macromodular Computer Systems", AFIPS Spring Joint Computer Conference, Vol. 30, pp. 335-336, 1967.

Ornstein, S. M., Stucki, M. J., and Clark, W. A., "A Functional Description of Macromodules", AFIPS Spring Joint Computer Conference, Vol 30, pp. 337-364, 1967.

[3] Falkoff, A. D., Iverson, K. E., and Sussenguth, E. H., "A Formal Description of System/360", IBM System Journal, Vol. 3, No. 3, pp. 198-263, 1964.

Chu, Y., "An Algol-Like Computer Design Language", Communications of the ACM, Vol. 8, pp. 607-615, October 1965.

[4] Hellerman, H., *Digital Computer System Principles,* McGraw-Hill Book Co., New York, 1967.

[5] Kilburn, T., Edwards, D. B. G., Lanigan, M. J., and Sumner, F. H., "One-Level Storage System", IRE Trans. on Electronic Computers, Vol. EC-11, No. 2, pp. 223-235, April 1962.

[6] Randell, B., and Kuehner, C. J., "Dynamic Storage Allocation Systems", Communications of the ACM, Vol. 11, No. 5, pp. 297-306, May 1968.

[7] Bell, C. G., "The Fundamentals of Time Shared Computers", Computer Design, pp. 44-59, February 1968 and pp. 28-46, March 1968.

[8] Goldstine, H. H., and von Neumann, J., "On the Principles of Large Scale Computing Machines", *John von Neumann Collected Works,* Vol. V, Pergamon Press, pp. 1-32, 1963.

Burks, A. W., Goldstine, H. H., and von Neumann, J., "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", Part I, Vol. 1, *John von Neumann Collected Works,* Vol. V, Pergamon Press, pp. 34-79, 1963.

Burks, A. W., Goldstine, H. H., and von Neumann, J., "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", Part I, Datamation, Vol. 8, No. 9, pp. 24-31, September 1962.

Burks, A. W., Goldstine, H. H., and von Neumann, J., "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", Part II, Datamation, Vol. 8, No. 10, pp. 36-41, October 1962.

[9] Buckholz, W., "The System Design of the IBM Type 701 Computer", Proc. of the IRE, Vol. 41, No. 10, pp. 1262-1275, October 1953.

[10] Lonergan, W., and King, P., "Design of the B 5000 System", Datamation, pp. 28-32, May 1961.

[11] Thornton, J. E., "Parallel Operation in Control Data 6600", AFIPS Fall Joint Computer Conference, Vol. 26, Part II, pp. 33-40, 1964.

[12] "The Structure of System/360", IBM System Journal, Vol. 3, No. 2, 1964.

[13] Chu, Y., *Digital Computer Design Fundamentals,* McGraw-Hill Book Co., New York, 1962.

[14] Flores, I., *Computer Design,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1967.

[15] Braun, E. L., *Digital Computer Design,* Academic Press, 1963.

[16] Meyerhoff, A. J., editor, *Digital Applications of Magnetic Devices,* John Wiley and Sons, Inc., 1960.

[17] Schuur, C. C. M., "A Fast 2½ D Mass Memory", Proceedings of SJCC, pp. 267-274, 1968.

[18] Frankel, S. P., "The Logical Design of a Simple General Purpose Computer", Trans. of the IRE PGEC, March 1957.

ACKNOWLEDGMENT

We wish to express our appreciation to the following individuals who contributed to the discussion that resulted in this report:

Gene Amdahl, IBM
H. Ferguson, Control Data Corporation
Martin Graham, University of California, Berkeley
William McKeeman, University of California, Santa Cruz
Kay Magleby, Hewlett Packard
Richard Mattson, IBM
Ralph Smith, Stanford University
Harold Stone, Stanford Research Institute