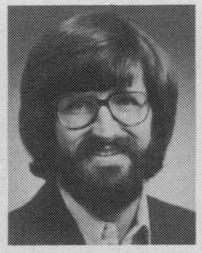
to be a good start for those that are. The microprocessor equipment described here has been used for several senior design projects. The AIM-65 system, along with the solderless interface, allows for easy testing and debugging of microprocessor-based senior design projects.

Acknowledgment

The author would like to express his appreciation to R. Cronin and K. Pruett for their help in the construction of the laboratory apparatus.

References

[1] R. C. Camp, T. A. Smay, and C. J. Triska, *Microprocessor Systems Engineering*. Portland, OR: Matrix, 1979, Appendix A.
[2] M. L. De Jong, "A simple 24 hour clock for the AIM-65," *Micro*, p. 10:5, Mar. 1979.
[3] D. F. Hanson, "Microcomputers in electrical engineering education at the University of Mississippi," in *Proc. 2nd Annu. Symp. Microprocessors and Education*, Huntsville, AL, Dec. 9, 1978, pp. 1-3-1-6.
[4] P. R. Rony, D. G. Larsen, and J. A. Titus, "Appendix 4: Outboards," in *The Bugbook V*. Derby, CT: E & L Instruments, Inc., 1977.



Donald F. Hanson (S'67–S'71–M'75) was born in Urbana, IL, on March 5, 1946. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, in 1969, 1972, and 1976, respectively.

For the 1976–1977 school year, he was an Assistant Professor with the Department of Electrical Engineering, Iowa State University, Ames. He is currently an Assistant Professor with the Department of Electrical Engineering, University of Mississippi, University. His research interests include the development of mathematical and numerical techniques for solving boundary-value problems of electromagnetic theory, and digital and analog electronics applications including microprocessors and microcomputer interfacing.

Dr. Hanson is a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu.

A Course Sequence in Microprocessor-Based Digital Systems Design

TREVOR MUDGE, MEMBER, IEEE

Abstract—**This paper gives an overview of a sequence of three courses designed to educate students in the use of microprocessor-based digital system design. The first course in the sequence is intended to satisfy the educational needs of undergraduates whose main focus of interest is not digital system design. The complete sequence is intended to provide a firm background in microprocessor-based digital system design for students who planned to continue in our graduate program and whose graduate work would involve them in experimental computer science. The three courses are part of a larger commitment to revitalize experimental computer science in the Department of Electrical and Computer Engineering at the University of Michigan. An overview of the lab equipment is given. The lab experiments are outlined and their pedagogical purpose briefly discussed.**

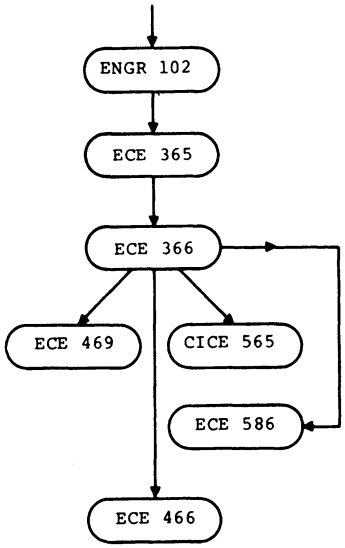
I. Introduction

THIS paper gives an overview of a sequence of three courses designed to educate students in the use of microprocessor-based digital systems design. These courses are presently being offered by the Electrical and Computer Engineering (ECE) Department and the graduate program in Computer, Information, and Control Engineering (CICE) at the University of Michigan. Design is taught in the context of microprocessors and other related very large scale integrated (VLSI) components, including ROM and RAM memory chips, PLA's, timers, sequencers, PIO's, SIO's, multipliers, and floating point attached processors. These VLSI components are taken principally from the Zilog Z80 family, the Advanced Micro Devices Am2900 family, and the Intel 8086 family. Fig. 1 shows a diagram of the relevant course sequence, including related courses. The three courses on which this discussion centers are: ECE 365—Digital Computer Engineering, ECE 366—Digital Computer Engineering Laboratory, and ECE 466—Digital Design Laboratory. ECE 365 is intended for juniors, ECE 366 for seniors, and ECE 466 for seniors or first year graduate students. Alongside the course sequence diagram is the catalog description of each course and the credit it is worth in hours. The direction of the arrows in the course sequence shows the order in which the courses must be taken—viewed in reverse the arrows define the prerequisite courses for each course.

Originally, in the period 1968 through 1978, 365 and 366 were taught using five PDP8 minicomputers—four for ECE 365 and one for 366—and a number of attache case "logic lab" kits that contain a 5 V power supply, push buttons, switches,

Manuscript received July 31, 1980.
The author is with the Computer, Information, and Control Engineering Program and the Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109.

0018-9359/81/0200-0014$00.75



ENGR 102. *Digital Computing* (2)*–Introduction to digital computer systems, flow diagrams, algorithms and digital computer programming. Design, implementation, and analysis of computer programs for numeric and nonnumeric computation.

ECE 365. *Digital Computer Engineering* (4)–Internal organization of digital computers; basic digital logic; control processor organization; input/output devices; interrupt facility; data channels; storage devices; machine language; studies of existing systems. Lectures and laboratory.

ECE 366. *Digital Computer Engineering Laboratory* (2)–Introduction to the practical aspects of designing digital computers using modern electronic components. Experiments with specially designed laboratory facilities. Lecture and laboratory.

ECE 466 (CICE 466). *Digital Design Laboratory* (2)–Realistic design problems in digital system engineering. Design, construction, and demonstration of devices which operate alone or in conjunction with digital computers in the laboratory. Lecture and laboratory.

ECE 469 (CICE 469). *Application of Real-Time Computer Systems* (4)–Principles of application of real-time computer systems to engineering problems. Topics include: computer characteristics needed for real time use, mini/macro computer operating systems, man-computer communication, basic digital logic design, analog signal processing and conversion, and intercomputer communication. Topics investigated via laboratory using microprocessor system. Three lectures and one three hour laboratory per week.

CICE 565. *Logical Design of Digital Computers* (3)–Advanced course in logical design. Logical properties of devices. Architectural properties, formal descriptions, and register-transfer simulations of digital systems. Control strategies including micro-programming. Hardware technology and its relationship to computer structures. Design automation of digital systems.

ECE 586 *Digital Circuits Laboratory* (2)–Design, characterization, and application of integrated electronics in projects utilizing optoisolators, phase-locked loops, logic, memory, and a variety of digital and analog signal processing. Applications of microprocessors. Present approaches to solid-state circuits in communications, instrumentation, sensors, and displays. Students select two or three projects. Laboratory work plus directed readings.

ENGR Engineering
ECE Electrical and Computer Engineering
CICE Computer, Information and Control Engineering

*2 hours worth of credit towards a baccalaureate requiring 128 hours credit.

Fig. 1. Relevant courses.

built-in sockets for IC's, wire, wire strippers, pliers, logic probes, and a selection of TTL logic. Assembly programming and logic design were introduced in 365, and I/O programming and projects involving interfacing hardware to PDP8's were introduced in 366.

During the first part of the same period 466 was an infrequently taught lab course in advanced digital design projects. In the mid 1970's 466 was revived and updated to include projects involving PDP16 register transfer modules (RTM's) as well as the then new Intel SIM8-01 system–an Intel 8008-based single board computer [1]. Furthermore, a link was established to MTS (Michigan Terminal System–the campus wide computing facility) where utilities such as cross assemblers, editors, etc., were available. This allowed rapid software development outside the lab for projects involving RTM's or the SIM8-01. This reduced the time needed in the lab, allowing more efficient use of the lab resources. The success of this link prompted the development of similar facilities for 365 and 366.

By 1977 the demand for 365 and 366 had grown, and both courses, which still relied on PDP8's, were very much outdated. It was felt that the courses needed updating and that modern microprocessor-based equipment was needed in their labs. Therefore, a proposal [2] to fund new lab equipment was submitted to the National Science Foundation (NSF) under their Instructional Scientific Equipment Program. It was successful, and together with matching funds from the University of Michigan put about $29 000 at our disposal to develop a "microprocessor-based computer and digital systems design laboratory." The proposal called for 12 lab stations to be built by us and shared by both 365 and 366. Each station was to be constructed around a Motorola MEK 6800D2 kit, a 16K X 8-bit RAM board, a low-cost CRT keyboard terminal, and a cassette tape recorder. These lab stations were to be used in conjunction with the logic lab kits. However, a gift of $30 000 from the XYCOM Corp. of Saline, MI allowed us to acquire ready-made microprocessor development systems for the lab stations to be used in conjunction with the logic lab kits. As a result, ten of XYCOM's Z80-based development systems–RacPac 3905's–were purchased and introduced into 365 and 366 during 1979. In addition, three printers were obtained–two Okidatas and one DEC Centronics. We considered the RacPac's to be a better choice for lab stations than our proposed home built unit as it had much more capability (see later), they were tried and tested, they were in industrial quality packages, and they came complete with the normal complement of software for a development system. These advantages allowed us to get the labs operational much more rapidly than would have been the case if we had had to stay with our original proposal. A disadvantage of our choice is that we now have ten instead of 12 lab stations. However, funds have recently been provided by the ECE Department to purchase two more RacPac's by the Fall semester, 1980.

In 1979 it was felt that 466 had also become dated. Therefore, funds were raised within the department to develop four state-of-the-art lab stations for 466. Building the stations was begun by 466 students as an in-class project. Hardware development was completed during the Winter semester, 1980.

TABLE I
YEARLY ENROLLMENT FIGURES

Course	Data					Estimates		
	1975	1976	1977	1978	1979	1980	1981	1982
ECE 365	185	192	208	236	245	326	350	350
ECE 366	68	71	70	67	96	114	120	120
ECE 466	49	44	47	48	51	54	60	60

Hardware validation and software development are planned for the Fall semester, 1980. The stations are based on the Intel SDK86 evaluation kit, which includes an 8086 microprocessor and Am2900 bit slice components. We have termed the stations 29/86 systems. Larger logic lab kits have also been built to be used in conjunction with the 29/86 systems.

Table I shows how enrollments have grown in the last five years and the projected enrollments for the next three years. In spite of this growth we have actually been able to increase the number of hours each student has in contact with a lab station in both 365 and 366. This was achieved by increasing the number of stations from the PDP8 days, and by opening the lab for longer hours. In the case of 365 and 366 a single lab is used containing all ten (eventually 12) RacPac's. The lab is open 80 hours/week allowing each student in both classes about 3.75 (eventually 4.5) hours/week contact time. This is up from the 1.5 hours allowed when the PDP8's were used.

In reorganizing the class material of the courses to match the new lab equipment two aims were foremost. First, we wanted 365 to satisfy the educational needs of undergraduates whose main focus of interest was not digital system design. Second, we wanted the sequence 365/366/466 to provide a firm background in microprocessor-based digital system design for students who planned to continue in our graduate program and whose graduate work would involve them in experimental computer science. It has been noted in the widely quoted "Feldman Report" [3] that many of the significant early advances in computer science resulted from university-based experimental computer science projects. This report goes on to note the recent decline in experimental computer science at universities due to lack of funding, tenure pressures inconsistent with experimental work, and the flow of talented faculty and students from universities to industry. We are making a major effort to revitalize and strengthen experimental computer science within the ECE Department at the University of Michigan as evidenced by our development of the Computer and Image Processing Research Network (CIPRNET) with major funding from NSF over three years. A component of the CIPRNET is the ADDEPT (Application-Directed Digital Equipment Prototype Testbed) laboratory which has been created specifically to support simultaneous development of specialized software, firmware, and hardware using state-of-the-art VLSI components. The course sequence 365/366/466 is intended to contribute to the strengthening of our experimental computer science program by preparing students to make full use of the ADDEPT lab research facility by giving them hands-on instruction in the use of state-of-the-art VLSI components.

Before going on to discuss the courses in more detail a few comments are in order. As noted, the original versions of the three courses formed a basis for the new versions. In this regard the introduction of new lab equipment and more modern class material can be viewed as evolutionary. No new courses were created. This was important for two reasons: First, present faculty teaching loads make it difficult to staff new courses; second, the same faculty were used to teach the new versions of the courses which simplified retraining. The courses are discussed in more detail in the following sections.

II. DIGITAL COMPUTER ENGINEERING—ECE 365

This class is intended for students in their junior year. As noted in the previous section, this course is intended as a foundation course for students in the area of microprocessor-based digital system design as well as a service course for those students whose main focus of interest is in other areas of electrical engineering. This last requirement means the course has to be self contained. The catalog description is shown in Fig. 1. There are four hours of classes/week not including labs. The course class material centers on digital computer organization, assembly language, and logic design, with particular reference to microprocessor-based systems. Concepts in the classes are reinforced by a series of eight carefully chosen lab experiments that use the RacPac systems and logic lab kits. The course lecture material is in contrast to a more traditional course on digital system design in two important ways.

First, emphasis is placed on using microprocessors and other VLSI components such as PLA's, ROM's, RAM's, PIO's, SIO's, and timers as system components. Gate level logic design is discussed in much less depth than is usual in a more traditional treatment. Topics such as minimizing of combinational logic and state reduction in sequential machines are only briefly mentioned, whereas design techniques that use PLA's and ROM's are emphasized. Finding a textbook to reflect the thrust of the course has been a problem. Until now only the Mostek *Microcomputer Data Book* [4] has been used. This documents the Z80 family of VLSI components. However, when it becomes available we plan to add Peatman's *Digital Hardware Design* [5] to the book list.

Second, not only is assembly language taught but so is the use of the RacPac operating system, editor, loader, and other utility programs used in system development on the RacPac. This represents a considerable increase in the amount of software taught compared to that taught in a traditional treatment of digital system design. Our experience so far has shown this extra software to be an obstacle to covering the range of material that we feel constitutes a minimum for such a course. Assembly language appears to give the most difficulty to students. Therefore, we have started to develop some teaching aids to help students learn assembly language more rapidly. Chief among these is an interpreter for the Z80 assembly language. Development of this interpreter has been funded by the University of Michigan's Center for Research on Learning and Teaching. If development continues on schedule the interpreter will be introduced in the Fall semester, 1980. The interpreter disassembles specified segments of memory and

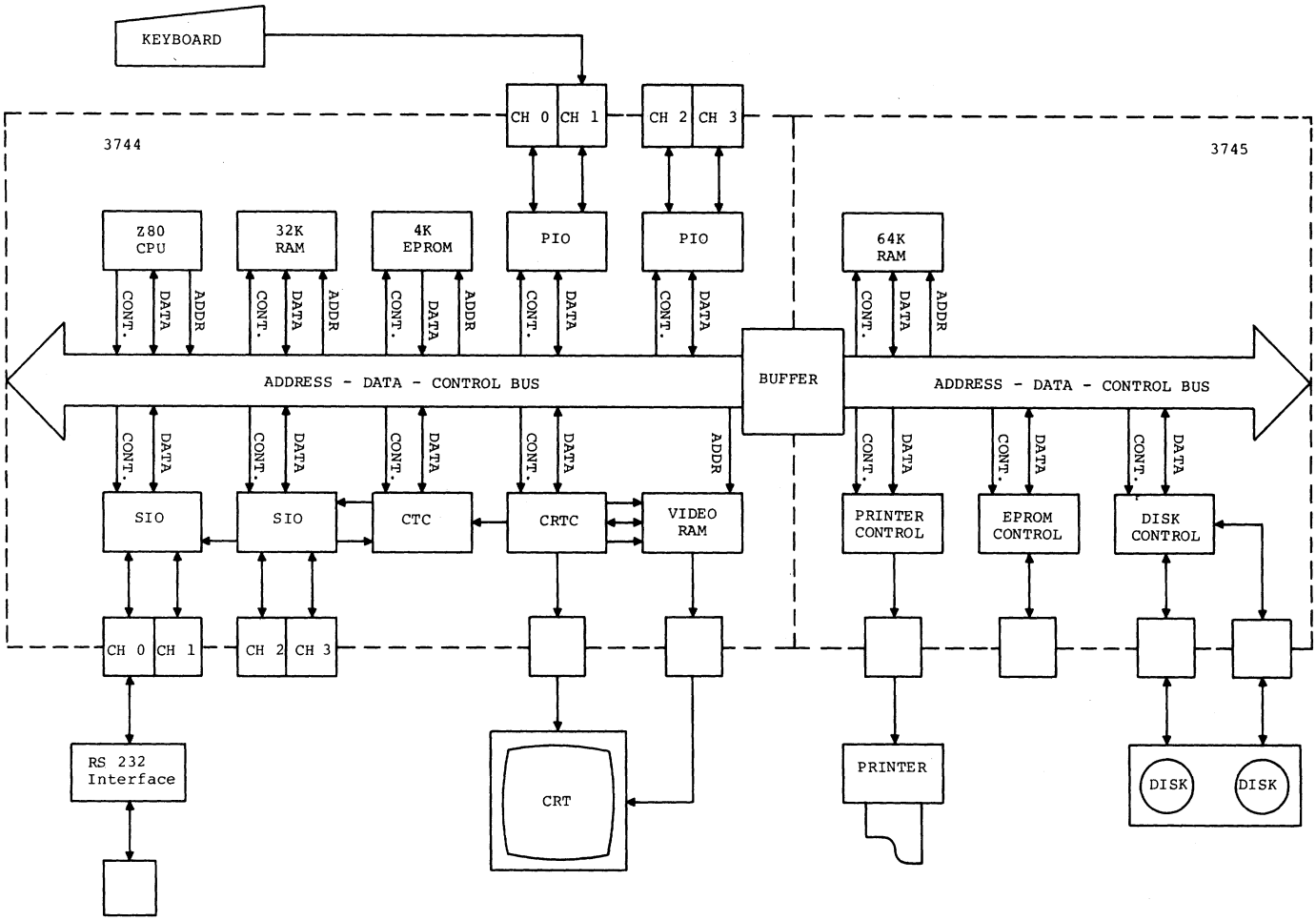


Fig. 2. RacPac.

displays the corresponding code with symbolic operation codes and absolute addresses. Interpretation of the code can be performed one instruction at a time or in normal sequence until a specific "trigger" condition is met. The instruction under interpretation is displayed in inverse video to distinguish it from the other code displayed. The CPU registers are always displayed along with the status flags. When an instruction is interpreted, any memory locations that are changed are displayed and any CPU registers or flags that are changed are displayed in inverse video to distinguish them from the unchanged ones. Interpreting single instructions clarifies their operation for the novice. Interpreting a program enables the user to relate the program's text (static) with the program's execution (dynamic). This is probably the single most difficult relationship that the novice must learn to visualize. We plan to use the interpreter in conjunction with in-class CRT monitors to explain instructions and to illustrate assembly language concepts. Students will also get a copy of the interpreter for use as a self-teaching aid and as a debugging tool. The "trigger" concept, which was borrowed from logic analyzers, makes the interpreter a particularly powerful debugging tool. Blocks of code can be executed until a trigger condition or a simple logical combination of trigger conditions becomes true. The conditions are specific values of locations, registers, or flags.

In the following two subsections the RacPac system and the lab experiments are described.

A. The RacPac System

The RacPac system is shown in Fig. 2. It is comprised of a CRT, a keyboard, dual single density Shugart 8 in disk drives, an RS232 interface, and connections for a printer and an EPROM burner. The logic is built on two boards designated 3744 and 3745. The 3744 (CPU) board contains the Z80 CPU, 32K X 1-byte dynamic RAM, 4K X 1-byte EPROM, two parallel input/output (PIO) chips, two serial input/output (SIO) chips, a counter/timer chip (CTC), a CRT controller (CRTC) chip, and a 2K X 1-byte static RAM. The 3745 (expansion) board contains 64K X 1-byte dynamic RAM, a floppy-disk controller chip, EPROM burner control logic, and parallel printer control logic.

The CPU is a Z80A-CPU chip, dynamic RAM is made up of 16K X 1-bit Motorola MCM4116 chips, EPROM is made up of 2K X 4-bit MCM2716 chips, and static RAM is made up of 1K X 4-bit MCM2114 chips. The PIO's are Z80A-PIO chips containing two channels each, the SIO's are Z80A-SIO chips containing two channels each, the CTC is a Z80A-CTC chip containing four channels, the CRTC chip is Motorola's MC6845, and the floppy-disk controller chip is Western Digital's FD 1771.

The keyboard is connected to PIO channel 1 (CH1). PIO channels 0 and 2 are available on back plane connectors. The RS232 interface is connected to SIO channel 0. The disk controller chip controls both disks. The printer and EPROM burner may be connected to their respective controllers through back plane connectors. The static RAM holds the CRT screen image and is controlled by the CRTC chip. Video timing is provided by CTC channel 3. The EPROM contains the bootstrap loader and some primitive operating system functions. The address space of the Z80 is 64K bytes. The lower 32K are occupied by 32K of dynamic RAM and the 4K of EPROM which can be banked switched with the low 4K of RAM. The upper 32K is occupied by either of two 32K banks of dynamic RAM and the 2K static video RAM which can be banked switched with the high 2K of whichever 32K bank of dynamic RAM occupies the upper half of the memory space. The video RAM's control is time multiplexed between the CRTC and the CPU so that screen refresh and screen updates from the CPU can continue with apparent concurrency.

The next subsection details the lab experiments.

B. The Lab Experiments

Experiment 1–Introduction to the RacPac: This experiment is designed to familiarize the student with the basics of using the RacPac system. The experiment write-up gives the student detailed step-by-step instructions for switching on the system, loading the system, and using the floppy-disk operating system and editor in creating a program. The write-up assumes that students have no knowledge of computers beyond an understanding of number systems in binary, octal, and hexadecimal. A program in Z80 assembly language is included in the write-up which performs a bubble sort. Students are required to assemble, link, and run this program. They must also demonstrate it by showing it performing an in-place sort on 256 bytes of memory. This can be done by displaying the appropriate 256 bytes of memory using the operating system before and after the execution of the program.

This experiment is intended to illustrate the rudiments of the operating system, editor, and the assembly and linking procedure.

Experiment 2–Simple Input/Output Programming: This experiment assumes that the relevant points about the Z80 instructions have been covered in the preceding classes. The requirement is to write a program that indicates whether an alphabetic or numeric key has been pressed by generating an appropriate message on the CRT.

The experiment is intended to develop the concepts of reading from and writing to I/O devices. The students are encouraged to write their I/O routines as subroutines. This requires them to become familiar with the stack mechanism in the Z80. Debugging techniques are introduced using the breakpoint mechanism of the operating system.

Experiment 3–ASCII to Hexadecimal Conversion: The program supplied in Experiment 1 sorts bytes (pairs of hex digits). Its operation is observed by using the operating system to view the pertinent 256 bytes of memory before and after it is run. Experiment 3 requires the students to write I/O routines that allow the bubble sort to be used from the console.

The I/O routines require the students to devise techniques for conversion between the ASCII characters, 0, 1, $\cdots$, 9, A, $\cdots$, F, and their hex equivalent. The bubble sort is called as a subroutine.

Experiment 4–Bucket Sort: This experiment requires students to write a bucket sort (a type of radix sort–see [6]) to replace the bubble sort subroutine used in Experiment 3. The speed of the two sorts are then compared using one of the RacPac's timer channels. Software to run the timer channel is part of the software package each student copies from the lab master disk at the beginning of the course. Plots are made of number of items to be sorted versus time for both sort subroutines. The number of items range from 20 to 256. The whole experiment is repeated on presorted data.

The purpose of the experiment is to introduce the student to the importance of algorithmic efficiency (in this case time efficiency). The student should observe the bubble sort times growing approximately quadratically with respect to the number of items, and the bucket sort times growing linearly with respect to the number of items. By contrast, in the second phase of the experiment the student should observe the bubble sort performing faster than the bucket sort, whose performance is related only to the number of items to be sorted, not the initial order of the data.

Experiment 5–Multiplication: This experiment requires the student to write a program to accept two 8-digit unsigned numbers from the keyboard, display them on the CRT, then compute their 16-digit product and display it.

To accomplish this the student must develop a multiword BCD shift and add multiply routine. The student is encouraged to solve the problem of digit-wise multiply by table lookup, where the multiplication table is generated during the initialization phase of the program. Software to check and time the multiplication is available from the master disk.

Experiment 6–Adders and Flip-Flops: This experiment is the first using the logic lab. It is assumed that the relevant theory has been covered in the preceding classes. Students are required to build a full adder and various types of flip-flops using only NAND gates.

Experiment 7–T-bird Turn Signals (Logic): This experiment again uses the logic lab, and requires the student to develop sequential logic to perform the familiar "rippling" T-bird turn signals used on the Ford Thunderbirds in the 1960's. Six LED's in the logic lab are used for the turn signal lights and a three position switch in the logic lab is used to input direction–left turn, right turn, or no turn. Additionally, a two position switch is required as a "hazard" switch. When it is on it should override the turn signal sequence and cause all the turn lights to flash repeatedly.

The design procedure recommended is based on the algorithmic state machine (ASM) concept developed at Hewlett-Packard (see [7]). As a starting point and as part of the experiment write-up a flowchart defining the ASM for the problem is given to the students. They are required to realize the ASM using MSI level components such as counters, multibit latches, multiplexers, and a register file to store next state information.

In the near future we intend to modify this experiment to

use EPROM's to store next state information instead of the volatile register files. We presently have two RacPac compatible EPROM burners, only the necessary EPROM's need to be purchased.

Experiment 8–T-bird Turn Signals (RacPac): This experiment repeats Experiment 7 but implements the same ASM by a program running on the RacPac. The switches are sampled by the Z80 and appropriate light patterns are output to the LED's in the logic lab.

The purpose of this experiment is to illustrate how microprocessors can be used to replace "one off" logic designs, especially in slow speed applications. This is an important design concept that is emphasized in the course class material. It is the first step in developing the concept of the microprocessor as a component in the design of a digital system.

III. Digital Computer Laboratory–ECE 366

This class is meant for students in their senior year. It is a follow-on from 365 and is predominantly a lab course–there is only one hour of formal class/week. The lab experiments are carried out on the RacPac systems and the logic lab kits. Most lab experiments involve building some logic in the logic lab kits and interfacing it to a RacPac. The logic is usually of sufficient complexity that it cannot be built and tested in one sitting, therefore each student is assigned a logic lab kit for the duration of the course. To help with debugging, three logic analyzers are available to the students. These are an HP1615A and two HP1602A's. These were donated by Hewlett-Packard in 1979. The logic analyzer has rapidly become the major diagnostic tool in the construction of experimental digital systems, replacing the oscilloscope. It is thus important that students become familiar with these tools if they are to have a balanced education in digital system design. With this in mind we have encouraged their use in the lab by discussing in each lab write-up how they may be used to advantage in the corresponding experiment. Our experience has shown that after a brief initial learning phase the benefits of using the logic analyzers pays off in allowing students to deal more efficiently with complex digital systems than was possible when just oscilloscopes were used.

Below follows a more detailed description of the lab experiments. There are six in all, and students are expected to attempt each one. Our eventual plan is to increase the number and develop a database of 20 or so experiments. Each student will then be able to select six from the database. The selection will represent a contract that the student must attempt to fulfill within the semester to receive a grade. The extent to which the student completes the contracted experiments will determine his final grade. No interim deadlines will be set–it will be up to the individual to set his own pace.

Experiment 1–Introduction to the Lab: This experiment is the same as Experiment 1 in 365 that familiarizes the student with the basics of using the RacPac system. For those students who have reached 366 through 365–the normal route–this experiment is repetitious. However, it gives transfer students and students who completed 365 before the RacPac systems were installed the chance to acquaint themselves with unfamiliar equipment before beginning more demanding experiments.

In addition to the RacPac experiment a simple logic design experiment must also be completed. This experiment is designed to allow the student to gain experience with the logic analyzers, particularly the complex 1615A. As a logic design exercise the experiment is trivial. A 4-bit counter must be built in two different ways. First, an SN7493A 4-bit ripple counter is wired up and set running. The 1615A is then used to display state information and timing information captured while the counter is running. An SN74154 4-line-to-16-line decoder is also used with the counter, and timing information about some of its lines is displayed. The ripple counter causes glitches to appear on some of these lines. These glitches are used to illustrate the glitch analysis capability of the 1615A. Second, a 4-bit ripple counter is constructed from four SN7476 flip-flops. The behavior of this second counter should be the same as the first. This is confirmed by making comparisons using the 1615A.

Experiment 2–A Full Adder: This experiment requires the student to build a full adder using two 4-line-to-1-line multiplexers. It is to be driven by the RacPac system through an uncommitted PIO channel.

Two programs are required to be written in connection with this experiment. The first is to verify that the full adder behaves in accordance with the truth table for a full adder. The second is to perform a bit serial byte add using the full adder. The PIO is operated under program control.

The intent of this experiment is to give the student experience in simple I/O under program control and to become more familiar with the operation of the PIO unit.

Experiment 3–Gunfighter: This experiment requires the student to devise the following game that determines which of two contestants has the faster reaction times: At the beginning of the game a 4-bit pattern is displayed on the RacPac's CRT. Subsequently, new 4-bit patterns are displayed one at a time separated by about 1/2 s. The sequence of 4-bit patterns is pseudorandom. Two push buttons form the input to the game. Each contestant gets one. When a contestant sees a 4-bit pattern that is the same as the one displayed at the start of game he/she should signal recognition by pushing the button. The winner is the first contestant to push his/her button when a correct match has occurred. The winning reaction time is displayed.

The push buttons should input to the RacPac as interrupts through an uncommitted PIO. The reaction time should be timed using an uncommitted CTC channel.

This experiment is intended to give the students experience with interrupt driven I/O.

Experiment 4–A 4 X 4 Multiplier: This experiment requires the student to construct a 4 X 4 multiplier in the logic lab that operates as an attached processor. The multiplier should be implemented as a shift and add algorithm using MSI TTL, in particular, the SN74181 4-bit ALU chip. The 4-bit multiplier and multiplicand should be regarded as unsigned numbers. An 8-bit product should be produced and input to the RacPac by interrupting through an uncommitted PIO channel.

This experiment is intended to extend the students experi-

ence with interrupt driven I/O and to develop logic design skills.

Experiment 5–Simulate a Data Communication Link: This experiment requires the student to simulate a data communication link using an uncommitted SIO channel and a 10-bit shift register.

The link should accept serial data from the SIO's transmitter and shift it through the 10-bit shift register (this delay simulates communication time). The shift register's output should drive the same SIO channel's receiver. An EXCLUSIVE OR gate in the middle of the shift register is to simulate noise in communication. The student is required to write the necessary software to transfer characters over the link and to correct single errors in communication.

The purpose of this experiment is to familiarize the student with the SIO's operation, and introduce him/her to some elementary aspects of data communication protocols.

Experiment 6–Direct Memory Access: This experiment requires the student to design and build logic to modify 16 words of memory by adding a 4-bit quantity to each of them. The words are to be retrieved for modification by direct memory access (DMA) and then returned to memory by DMA. The heart of the design problem is a bus controller.

The purpose of this experiment is to expose the student to the important technique of DMA and to become more familiar with bus protocols.

IV. Digital Design Laboratory–ECE 466

This class is intended for students in their senior year or first year of graduate school. It is a follow-on from 366 and is predominantly a lab course–there is only one hour of formal class/week. The intent of this class is to give students hands-on experience with state-of-the-art VLSI components by allowing them to carry out projects involving such components during the course of the semester. Students propose their own projects with guidance from the class instructor and after having spent some time familiarizing themselves with the lab facilities. Since admission to the class is contingent on the student having completed 365 and 366 or their equivalent, a reasonable level of sophistication in the area of digital systems design can be expected and, consequently, in the level of project that is appropriate. The basic lab station to be used in the projects is the 29/86 system mentioned in the Introduction and described in more detail below. Additional components will include the TRW TDC1010J 16 X 16 bit multiplier-accumulator chip, the recently announced Intel 8087 floating point processor, and the Fairchild 9400 family of chips which center on the 16-bit I3L microprocessor, the 9445.

As noted in the Introduction the 29/86 systems have not been completed. In fact only the hardware has been completed so far. A total of four stations were completed as an in-class project. Hardware validation and software development are planned for Fall semester, 1980 in-class projects. During this development phase the range of projects available to students is of necessity rather limited. However, if development continues on schedule the stations will be fully operational by Winter semester, 1981. At that time students will be able to devise their own projects within the guidelines mentioned above.

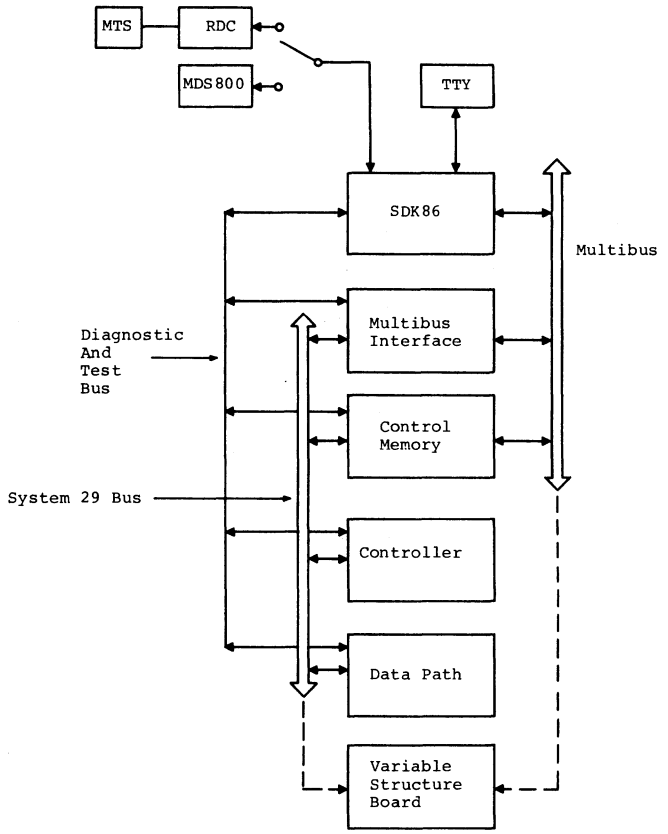


Fig. 3. 29/86 system.

We conclude this section with a brief overview of the 29/86 system.

A. The 29/86 System

A block diagram of the 29/86 system is shown in Fig. 3. The system occupies five boards housed in a chassis complete with power supply. The system partitions into two subsystems: the Intel SDK86 evaluation board containing the 8086 CPU, and the "System 29," an AM2900-based system, on the remaining four boards. The lab station is completed by a "variable structure board" (VSB) that is a larger version of the logic lab kits used in 365 and 366.

The SKD86 board is connected through one of its SIO ports to a manual switch that allows either a connection directly to MTS through a remote data concentrator (RDC) or a connection to an Intel 8080-based development system–the MDS800–augmented with dual 8 in double density disk drives. The MDS800 and the RDC connection are shared by all four stations. A second SIO port connects to a teletype. The SDK86 board connects to the System 29 through two buses. One of these is the diagnostic and test bus which connects a PIO on the SDK86 board to a pair of break point registers in the System 29. The other is the multibus from the SDK86 board. This is the bus structure supported by the Intel 8086 family of VLSI components (the SDK86 actually supports a reduced version of the multibus). The multibus connects to the System 29 bus through common memory (the control memory–

see Fig. 3) and also through an interface (the multibus interface) that allows the System 29 bus complete access to the multibus (but not vice versa).

The System 29 falls into four parts each on a separate board. First, there is the multibus interface to the System 29 bus. Second, there is the control memory which is organized as a 1K X 80-bit RAM and a 1K X 80-bit ROM (both can be increased to 2K). This forms the control store for the System 29. It outputs an 80-bit control word to the control memory data register (CMDR). It can be directly accessed from the SDK86 over the multibus, and to the SDK86 it appears as 10K of 16-bit memory. Third, there is the controller. The key item on this board is the Am2910 microprogram controller. This generates a 12-bit address to access one of the up to 4K control words in the control memory. It in turn is controlled by a field from the CMDR. Fourth, there is the data path. This is built around four Am2903 superslices and an Am2904 status and shift control unit. Together they form a 16-bit data path that includes a set of 16 registers and an ALU. The Am2903's are controlled by the same field of the CMDR. The control signals for the Am2904 do not change very often, so a residual control technique is used for it. The CMDR also includes a 16-bit immediate data field that allows immediate data to be placed on the data path and on the input to the microprogram controller.

Project dependent hardware can be built on the VSB; then the VSB can be interfaced to the System 29 bus to integrate the hardware into the System 29. If RAM modules are set up on the VSB and it is interfaced to the System 29 bus, the System 29 becomes a general purpose computer. Multibus compatible hardware such as the 8087 can be accessed by System 29 through the multibus interface.

In developing a project system the SDK86 can be used as a supervisor. Software for it can be developed on MTS or the MDS800 and downloaded. Software for the System 29 can also be developed on MTS or the MDS800 using AMDASM, AMD's microassembler that is supported on these two systems. The SDK86 can be used to load the control memory with the output of AMDASM. Using the diagnostic and test bus the SDK86 can set up a pair of break point registers to facilitate debugging the project system. The break point registers are compared to the CMDR and the operation of the System 29 halts when a match occurs (two registers are used when it is necessary to cover both outcomes of a branch).

V. Conclusion

We have described a sequence of three courses. The first is designed to be both a foundation course in microprocessor-based digital system design and a service course. The other two are designed to build on this foundation so that students completing the three course sequence are prepared for graduate work in experimental computer science.

Demand for all of the courses has been very heavy, and we have had to put a limit on their enrollments (see estimates in Table I). Also, the College of Engineering requires students to evaluate their courses using a computer form, and an open-ended form giving a suitable quantitative essay-like response. These responses have generally been very positive about the courses. The only major criticism has been the work load involved in taking the courses.

Copies of lab write-ups, a preliminary version of the interpreter, and details of the 29/86 system may be obtained from the author.

Acknowledgment

The author would like to thank all those who have helped in the development of the above courses, in particular, Prof. D. E. Atkins, C. M. Wilson, H. J. Keesom, A. B. Siegel, W. A. MacLeod, and B. Makrucki.

References

[1] J. B. Baron and D. E. Atkins, "An educational laboratory in contemporary digital design," in *Proc. 2nd Annu. Symp. Comput. Architecture*, Houston, TX, Jan. 1975.

[2] T. Mudge, "Microprocessor-based computer and digital systems design laboratory," Proposal to the National Science Foundation, Washington, DC, Grant No. SER78-13889, Sept. 1978.

[3] J. A. Feldman and W. R. Sutherland, "Rejuvenating experimental computer science," *Commun. Ass. Comput. Mach.*, vol. 22, pp. 497–502, Sept. 1979.

[4] *Microcomputer Data Book*. Carrollton, TX: Mostek Corp., 1979.

[5] J. B. Peatman, *Digital Hardware Design*. New York: McGraw-Hill, 1980.

[6] D. E. Knuth, *The Art of Computer Programming*, vol. 3, *Sorting and Searching*. Reading, MA: Addison-Wesley, 1975.

[7] C. R. Clare, *Designing Logic Systems Using State Machines*. New York: McGraw-Hill, 1973.

Trevor Mudge (S'74–M'77) was born in London, England, on November 28, 1947. He received the B.Sc. degree in cybernetics from the University of Reading, England, in 1969, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1973 and 1977, respectively.

Presently he is an Assistant Professor in the Department of Electrical and Computer Engineering (ECE) and with the Computer, Information, and Control Engineering Program, University of Michigan, Ann Arbor. He is a cofounder of the Application-Directed Computer Architecture Group in the ECE department, and has been involved in establishing their experimental computer architecture laboratory. His current research interests focus on the design of very large scale integrated circuits for highly parallel application-directed computer architectures. The application areas of emphasis are scientific computing and robotics. His teaching includes courses in the areas of computer architecture and software engineering, and he is engaged in the development of a state-of-the-art microprocessor laboratory. He is a consultant in the area of microcomputer-based systems design; in particular he is presently involved in specifying a next generation microcomputer architecture.

Dr. Mudge is a member of the British Computer Society, the Institution of Electrical Engineers, London, the Association for Computing Machinery, the Association for Computing Machinery Sigarch, and Sigma Xi.