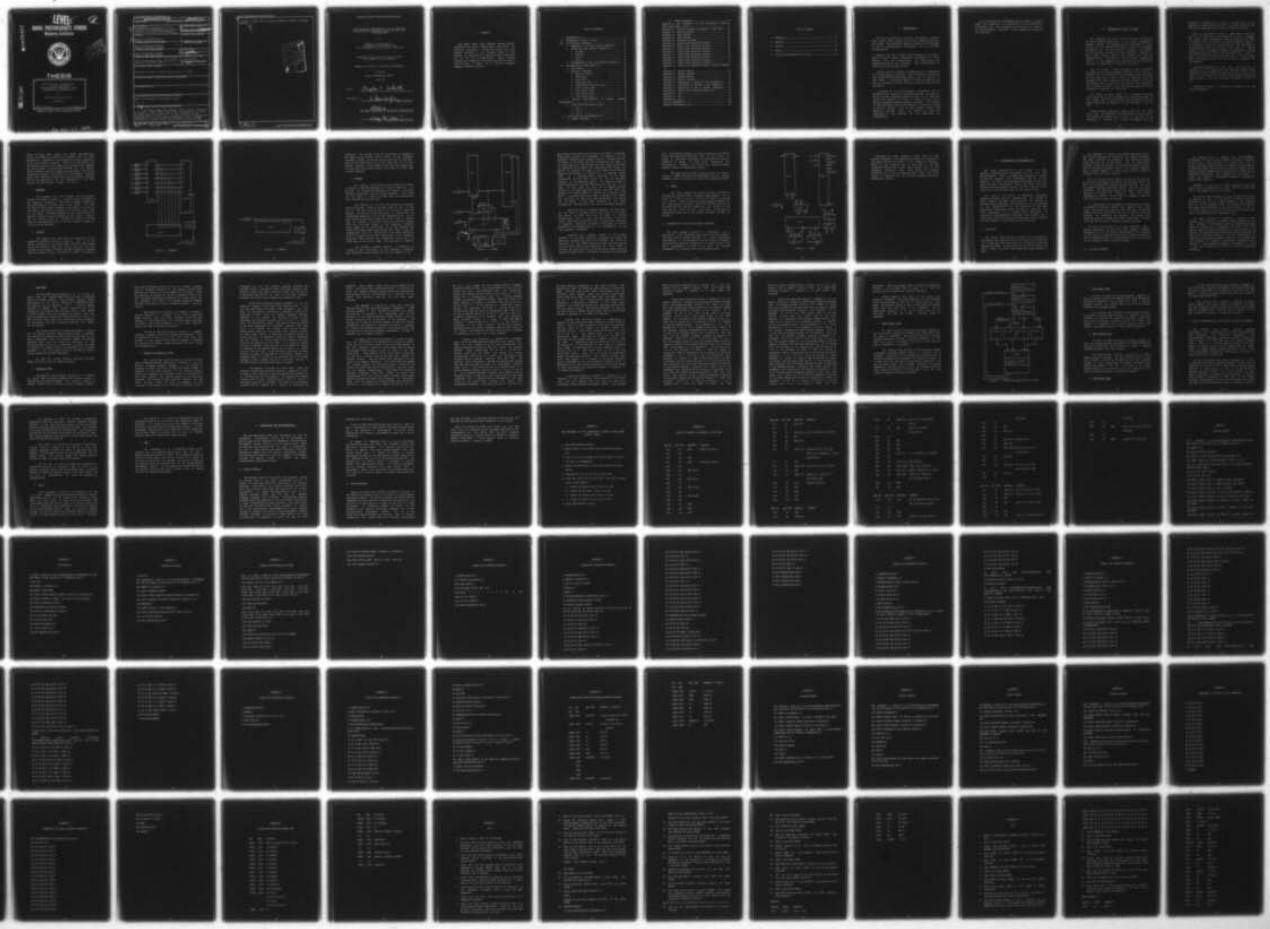
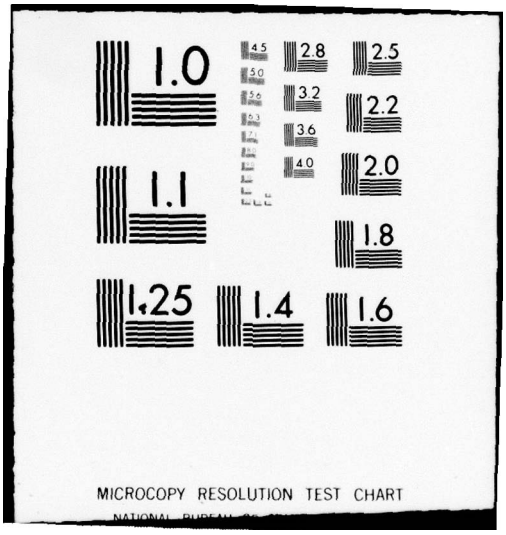
AD-A073 927 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
THE DESIGN AND IMPLEMENTATION OF AN INEXPENSIVE MICROPROCESSOR --ETC(U)
JUN 79 D F CORTEVILLE

UNCLASSIFIED NL

1 OF 2
ADA 073927




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART

LEVEL II

AD A073927

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DDC
SEP 19 1979

THESIS

THE DESIGN AND IMPLEMENTATION
OF AN INEXPENSIVE
MICROPROCESSOR DEVELOPMENT SYSTEM
FOR THE Z-80 MICROPROCESSOR

by

Douglas Floyd Corteville

June 1979

Thesis Advisor: R. Panholzer

Approved for public release; distribution unlimited.

DDC FILE COPY

79 09 17 096

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) The Design and Implementation of an Inexpensive Microprocessor Development System for the Z-80 Microprocessor		5. TYPE OF REPORT & PERIOD COVERED (9) Master's thesis; 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Douglas Floyd Corteville		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS (11) Naval Postgraduate School Monterey, California 93940		12. REPORT DATE Jun[illegible]79; 13. NUMBER OF PAGES 101
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California 93940 (12) 102 p.		15. SECURITY CLASS. (of this report) Unclassified; 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

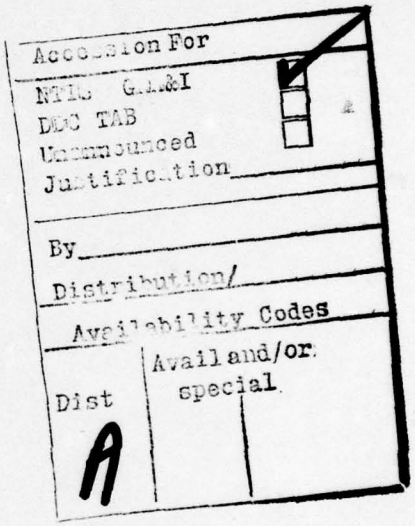
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Z-80, microprocessor development system.

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

The Radio Shack home computer system, TPS-80, configured with Level II Basic ROM's, 16K of RAM, its expansion interface, a single disk drive, and a line printer interface to the TELETYPE model 40 line printer is being used as a microprocessor development aid for the Z-80 microprocessor. Basic language programs are resident on the mini-disk and are used to load,

to store, to dump, and to execute assembled assembly language programs.



Approved for public release; distribution unlimited

THE DESIGN AND IMPLEMENTATION OF AN INEXPENSIVE MICROPROCESSOR DEVELOPMENT SYSTEM FOR THE Z-80 MICROPROCESSOR

by

Douglas Floyd Corteville
Lieutenant, United States Navy
B.S., Michigan State University, June 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

June 1979

Author: Douglas F. Corteville

Approved by: R Parkinson
Thesis Advisor

Chairman, Department of Electrical Engineering

William M. Tolles
Dean of Science and Engineering

ABSTRACT

The Radio Shack home computer system, TRS-80, configured with Level II Basic ROM's, 16K of RAM, its expansion interface, a single disk drive, and a line printer interface to the Teletype model 40 line printer is being used as a microprocessor development aid for the Z-80 microprocessor. Basic language programs are resident on the mini-disk and are used to load,to store, to dump, and to execute assembled assembly language programs.

TABLE OF CONTENTS

I. INTRODUCTION..... 8

II. CONVERSION TO LEVEL II BASIC..... 10

III. LINE PRINTER INTERFACE..... 12

A. OPERATION OF THE LINE PRINTER INTERFACE..... 12

B. INSIDE THE LINE PRINTER INTERFACE..... 13

1. Transmit..... 14

2. Receive..... 14

3. Control..... 17

4. Power..... 20

C. CONSTRUCTION OF THE LINE PRINTER INTERFACE... 20

IV. MICROPROCESSOR DEVELOPMENT AID..... 23

A. THE SYSTEM..... 23

B. THE BASIC PROGRAMS..... 24

1. Monitor Program..... 26

2. Dump Mode..... 27

3. Load Mode..... 28

4. Execution Mode..... 28

5. Single Step Execution Mode..... 29

6. Disk Storage Mode..... 35

7. Disk Recall Mode..... 37

8. Tape Storage Mode..... 37

9. Tape Recall Mode..... 37

C. SPECIAL CONSIDERATIONS FOR MACHINE LANGUAGE PROGRAMMING..... 38

D. USING THE DEVELOPMENT SYSTEM..... 40

1. Lab 1..... 40

2. Lab 2..... 41

3. Lab 3..... 42

V. CONCLUSIONS AND RECOMMENDATIONS..... 43

A. FUTURE SOFTWARE..... 43

B. FUTURE HARDWARE..... 44
Appendix A: THE PROCEDURE FOR THE CONVERSION TO LEVEL II BASIC FROM LEVEL I BASIC..... 46
Appendix B: THE 8748 PROGRAM TO GENERATE A LINE FEED... 47
Appendix C: MONITOR PROGRAM..... 52
Appendix D: DUMP PROGRAM..... 54
Appendix E: LOAD PROGRAM..... 55
Appendix F: EXECUTION PROGRAM..... 56
Appendix G: SINGLE STEP EXECUTION PROGRAM..... 57
Appendix H: SINGLE STEP EXECUTION PROGRAM 1..... 59
Appendix I: SINGLE STEP EXECUTION PROGRAM 2..... 60
Appendix J: SINGLE STEP EXECUTION PROGRAM 3..... 63
Appendix K: SINGLE STEP EXECUTION PROGRAM 4..... 65
Appendix L: SINGLE STEP EXECUTION PROGRAM 5..... 68
Appendix M: SINGLE STEP EXECUTION PROGRAM 6..... 71
Appendix N: SINGLE STEP EXECUTION PROGRAM 7..... 72
Appendix O: SINGLE STEP EXECUTION MACHINE LANGUAGE PROGRAM. 74
Appendix P: DSTORE PROGRAM..... 76
Appendix Q: DRECALL PROGRAM..... 77
Appendix R: TSTORE PROGRAM..... 78
Appendix S: TRECALL PROGRAM..... 79
Appendix T: SUBROUTINE 10 DECIMAL TO HEX CONVERSION.... 80
Appendix U: SUBROUTINE 100 DECIMAL TO HEX CONVERSION... 81
Appendix V: SUBROUTINE 200 HEX TO DECIMAL CONVERSION... 83
Appendix W: SINGLE STEP EXECUTION MEMORY MAP..... 85
Appendix X: LAB 1..... 87
Appendix Y: LAB 2..... 92
Appendix Z: LAB 3..... 96
LIST OF REFERENCES..... 100
INITIAL DISTRIBUTION LIST..... 101

LIST OF FIGURES

1. Transmit..... 15

2. Receive..... 16

3. Control..... 18

4. Power..... 21

5. Single Step Execution Flow Chart..... 36

I. INTRODUCTION

The inital condition of the Naval Postgraduate School's Radio Shack home computer was a Level I basic system with 16K of RAM. This beginning system was added to and modified to form a microprocessor development aid for the Z-80 microprocessor.

Chapter II will discuss the conversion of the Level I basic to Level II basic. This being complicated by Radio Shack's lack of documentation and instructions was accomplished within one afternoon. With instructions it would take about fifteen minutes

Chapter III will discuss the operation and construction of the line printer interface. This interface is controlled by an INTEL 8748 microprocessor and is set to operate at 2400 baud. A significant amount of time was spent on this due to a standard line feed character not being generated by the TRS-80.

In chapter IV the microprocessor development aid is discussed with application to sample problems. The use of a system of basic language programs resident on the mini-disk allows the operator to load, to dump, to store, and to execute assembled assembly language programs. The outputs of these operations are either directed to the video display or the line printer. Use of the system's single step execution program allows the operator to trace the operations on each register as the operations are accomplished.

The conclusions and recommendations of chapter V include ideas for furthering the usefulness of the TRS-80 as a development tool. The prospects of increasing the variety of microprocessors serviced by this inexpensive system is very exciting.

II. CONVERSION TO LEVEL II BASIC

The conversion to Level II basic from Level I basic was the first hardware change made to the Naval Postgraduate School's TRS-80. This changes the 4K of ROM used for Level I basic to 12K of ROM used for Level II basic. An increased instruction set and the software to support a line printer are the motivating factors for this alteration. Normally a Radio Shack service center would install and test the conversion kit. Since this is an electrical engineering thesis, it was decided that I would do the conversion and thus have an opportunity to study the CPU board.

The use of Ref. 1 during the Level II basic conversion is useful and the only available reference. All of the "Z " numbers are on the CPU board and therefore their locations are relatively easy to find. A new X3 is provided with the conversion kit and is used as is (DO NOT break any of the X3 connectors). The kit also has a 100 ohm 1/4 watt resistor. This resistor is used to load the recorder input but is not necessary; therefore, it is not used

The Level II basic ROM's are on a small 4-chip ROM board. This ROM board is attached with double-sided tape to the etched side of the CPU board. The ROM board has a flat ribbon cable used to connect it to the CPU board at Z33 or Z34

There are four lose wires coming from the ROM board which must be connectel to the CPU board: A yellow wire to be connected to address line 11 (A11); A red wire to be connected to address line 12 (A12); An orange wire to be

connected to address line 13 (A13); A green wire to be connected to something called ROM*. One must be careful for the color coding may change with different ROM boards.

ROM* is not defined in Ref. 1. ROM* must be high when A14 or A15 is high and is low otherwise, in order to support the memory map. Knowing what makes up ROM* leads to where it connects on the CPU board. On TRS-80 schematic (sheet 1) of reference 1, there is the DIP shunt X3. This DIP shunt is a shorting bar array. By breaking some bars and leaving others intact, the address decoder is programmed to reflect the amount of ROM and RAM on the CPU board. For a Level II basic ROM and 16K of RAM none of the bars of X3 are broken. ROM* is now found on pin 7 or 8 of X3 with an unbroken DIP is installed

The wires connecting to A11, A12, A13 must all be connected to the output of the tri-state buffers which isolate the CPU. Connect the yellow wire (A11) to pin 5 of Z37, the red wire (A12) to pin 13 of Z21, the orange wire (A13) to pin 3 of Z21, and the green wire (ROM*) to pin 8 of X3.

Appendix A contains a step-by-step procedure for the above conversion.

III. LINE PRINTER INTERFACE

To obtain hard copy of output data and program listings is an important part of any computer system. The TRS-80 expansion interface has two printer ports. One port is a screen printer port and interfaces with the Radio Shack screen printer. This port is a pin for pin copy of the 40-pin edge connector which connects the CPU board with the expansion interface. The other port is a line printer port which interfaces with the Radio Shack line printer. This port is a 32-pin port. It sends TTL data lines in parallel and receives four handshaking signals.

Given the need for hard copy output and the existence of a line printer which receives RS-232 signals, an interface was constructed to convert the TRS-80's expansion interface and the Teletype model 40 line printer.

A. OPERATION OF THE LINE PRINTER INTERFACE

The line printer interface must interface between the TTL parallel output and handshaking signals of the expansion interface and the RS-232, at 2400 baud, signals of the Teletype model 40 line printer. A universal asynchronous receiver/transmiter (UAR/T) the AY-5-1013 is used with a quad MDTL line driver the MC1488 to convert the parallel output of the expansion interface to 2400 baud RS-232 signals. A handshaking signal from the line printer is converted to TTL logic in a quad MDTL line receiver the MC1489.

The TRS-80 software does not generate an ASCII line feed character at the end of a line. This requires a smart line printer interface. An INTEL 8748 is used to control the handshaking signals, monitor the parallel output from the TRS-80, and when required take control of the printer to generate a line feed character. A line feed is generated after the TRS-80 transmits a carriage return.

Whenever the line printer interface and the disk drive are connected to the expansion interface, the line printer interface must be turned on. The lack of the proper handshaking signals, when the line printer interface is off, will prevent disk operation on start up and lock up the system during operation. Before the first use of the line printer interface after a shut down the RESET button must be pressed on the line printer interface to insure proper restart of the 8748.

B. INSIDE THE LINE PRINTER INTERFACE

Inside the line printer interface is divided into four parts. The transmit data path is discussed first. This is the path from the TRS-80 through the line printer interface to the line printer. The second part is the received data path which is the handshake signal from the line printer to the TRS-80. The control signals are generated next and pull together both transmit and receive signals along with the necessary clock pulses. The power supply of the line printer interface is examined for possible trouble shooting at a later date.

The line printer interface is made from three LSI and five SSI building blocks. One 8748 microprocessor, one AY-5-1013 UAR/T, one 8212 eight-bit input/output port, one

74126 tri-state quad buffer, one MC1488 quad MDTL line driver, one MC1489 quad MDTL line receiver, and two 7492 divide-by-12 counters are the integrated circuit components of the line printer interface. There are an additional six SSI building blocks on the expansion interface which support the line printer. These circuits decode the line printer driver address, store the output data character, receive the handshaking signals, and produce a 1/2 microsecond negative going data strobe to signal when data is present. Detailed data on the operation of all the components of the line printer interface appear in Refs. 2, 3 and 4.

1. Transmit

The transmit data path through the line printer interface is shown in figure 1. Parallel data is latched on the expansion interface. This data is passed through the 8212 configured as an output port to the UAR/T. The 8748 can read the data which is passed to the UAR/T or,as will be seen during the control discussion, the 8748 can write data to the UAR/T. The UAR/T changes the data from parallel to serial at the baud rate maintained by the control signals. The TTL serial output of the UAR/T is changed to RS-232 logic in the MC1488 and sent to the line printer through pin 3 of an RS-232 connector.

2. Receive

The receive data path from the line printer to the expansion interface is shown in figure 2. Pin 14 of the RS-232 connector is the receive next character (RNC) signal of the line printer. When this signal is low, the line printer is not ready to receive the next character. Likewise when this signal is high, the printer is ready to receive a

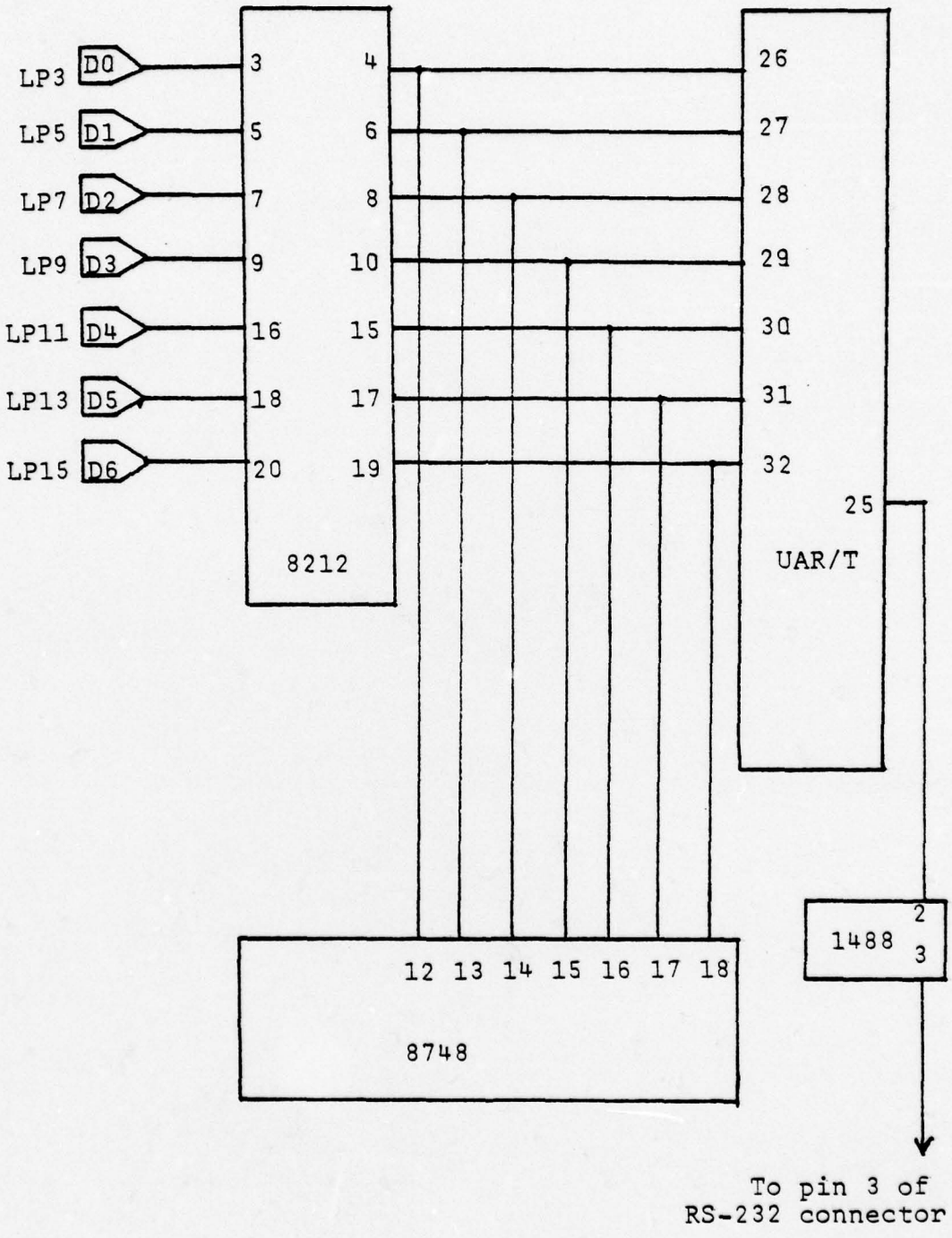


Figure 1 - TRANSMIT

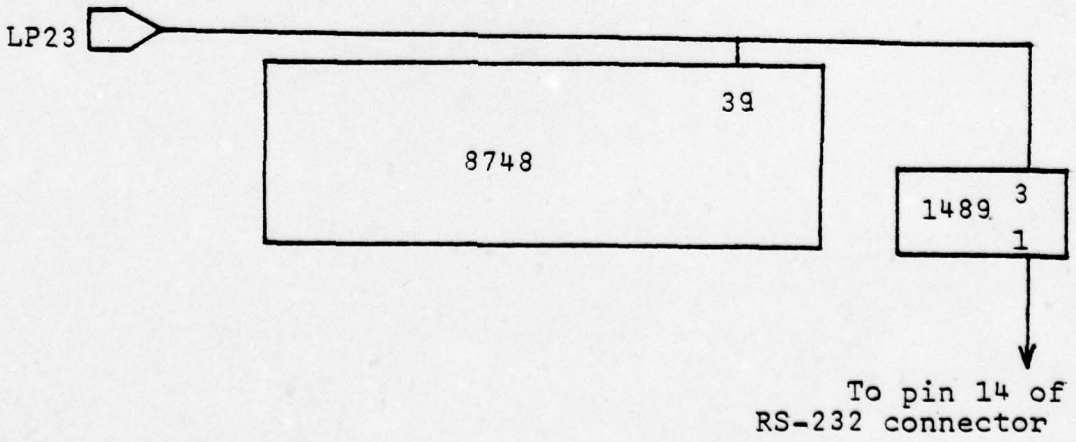


Figure 2 - RECEIVE

character. The RS-232 logic of the printer is changed to TTL logic by the MC1489. A low on LP-23 of the expansion interface causes the TRS-80 to halt until LP-23 goes high. This is also connected to T1 of the 8748 for use in the control sequence. The timing is such that this signal will not slow down the process unless a line feed or a form feed is being executed.

3. Control

The control of the line printer interface is shown in figure 3. The 8748 performs two control functions. It monitors the transmitted characters and inserts a line feed character after each carriage return character. The clock for the UAR/T, a clock at sixteen times the desired baud rate, originates from the 8748.

The control of the line feed character generation works as follows. Three of the data lines of port P1 and one of the data lines of port P2 of the 8748 are used for control. Initially, the BUS port, which is monitoring the transmitted character, is in a high impedance state. P12 is high, enabling the 8212 through one of the 74126's buffers used as a driver. P11 is low, enabling the TRS-80 through LP-21 and a tri-state buffer on the expansion interface which leads to D7 of the CPU. P13 is low, disabling the 74126's tri-state buffer which drives P20. P20 is high, but disconnected from the data strobe of the UAR/T (DS(not)). LP-26 is a clear line for the D flip-flops on the expansion interface which stores the transmitted character; it is wired high to disable the clear. The interrupt is enabled.

The transmit sequence follows. A character is latched to the D flip-flops on the expansion interface. This character passes through the 8212 and appears on the

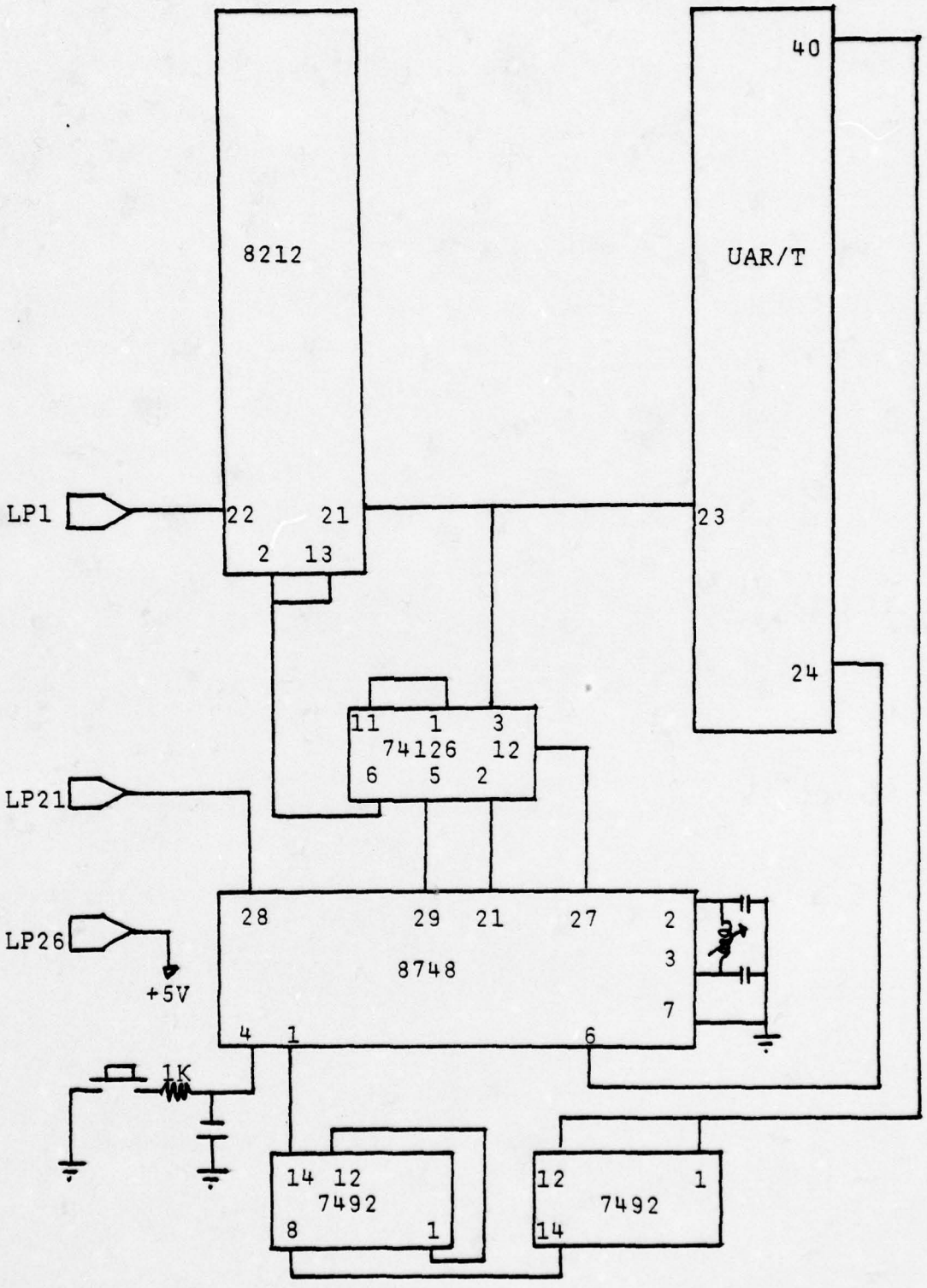


Figure 3 - CONTROL

data lines of the UAR/T and the BUS of the 8748. A negative going pulse of about 1/2 microsecond is generated on the expansion interface and is passed from LP-1 through the 8212 to DS(not) of the UAR/T. The end of character line (EOC) of the UAR/T goes low, interrupting the 8748, and the character is transmited by the UAR/T. The interrupted 8748 stops the TRS-80 by sending P11 high. It then inputs on the BUS the character and checks it to see if it is a carriage return (0DH). If a carriage return character is not being transmitted, then the 8748 starts its end of interrupt sequence (to be described later). If a carriage return is being transmitted, then the 8748 waits for the EOC to go high. At that time, P12 goes low to disable the 8212 (places it in a high impedance state) and P10 goes high enabling the 74126's buffer to connect P20 to DS(not) of the UAR/T. A line feed character (0AH) is places on the 8748's BUS. P20 goes low, then high, thus generating a data strobe for the UAR/T. A line feed character is now being transmitted. The 8748 starts its end of interrupt sequence.

During the end of interrupt sequence, the 8748 waits for the EOC to go high, then it waits for T1 to go high. This means the 8748 waits for the UAR/T to finish transmitting, then it waits for the printer to say it is ready. When these things are true, a return from interrupt is executed. After the interrupt is reenabled, the initial conditions are reestablished and the transmission of the next character is started.

Appendix B is a complete listing of the program resident on the 8748. Along with supporting the above, the 8748 also generates the clock for the UAR/T. The clock is generated by the 8748's instruction ENT0 CLK which places 1/3 the 8748's clock on the test line T0. This clock is divided by twelve and two by 7492 divide-by-twelve counters before the clock goes to the UAR/T at sixteen times the baud

rate. The variable inductor is used to fine tune the 8748's clock, thus fine tuning the 2400 baud of the UAR/T. To change to 4800 baud, disconnect the divide-by-two counter. Similarly to change to 9600 baud, disconnect the divide-by-two counter and rewire the divide-by-twelve counter to make a divide-by-six counter.

The reset button wiring is shown as part of control, figure 3. Before one tries to disassemble the line printer interface case, the reset button must be disconnected first.

4. Power

The power supply for the line printer interface is provided by a surplus radio telegraph power supply. It is rated at .5 amp at 5 volts, 125 ma at +12 volts, and 125 ma at -12 volts. Figure 4 shows the power supplies needed for the line printer interface. Pins 34-39 of the UAR/T set up the UAR/T for one stop bit, seven bits per character, and even parity. Pins 13 and 4 are high on the 74126 to permanently enable two of the 74126's tri-state buffers as drivers.

C. CONSTRUCTION OF THE LINE PRINTER INTERFACE

The line printer interface is constructed on a six-by-four inch perforated board using wire wrap sockets. The use of the wire wrap technique is a very significant contribution to the success of the line printer interface. The author has used wire wrapping on two other projects and never experienced a short or an open. He strongly recommends this technique for any prototype project.

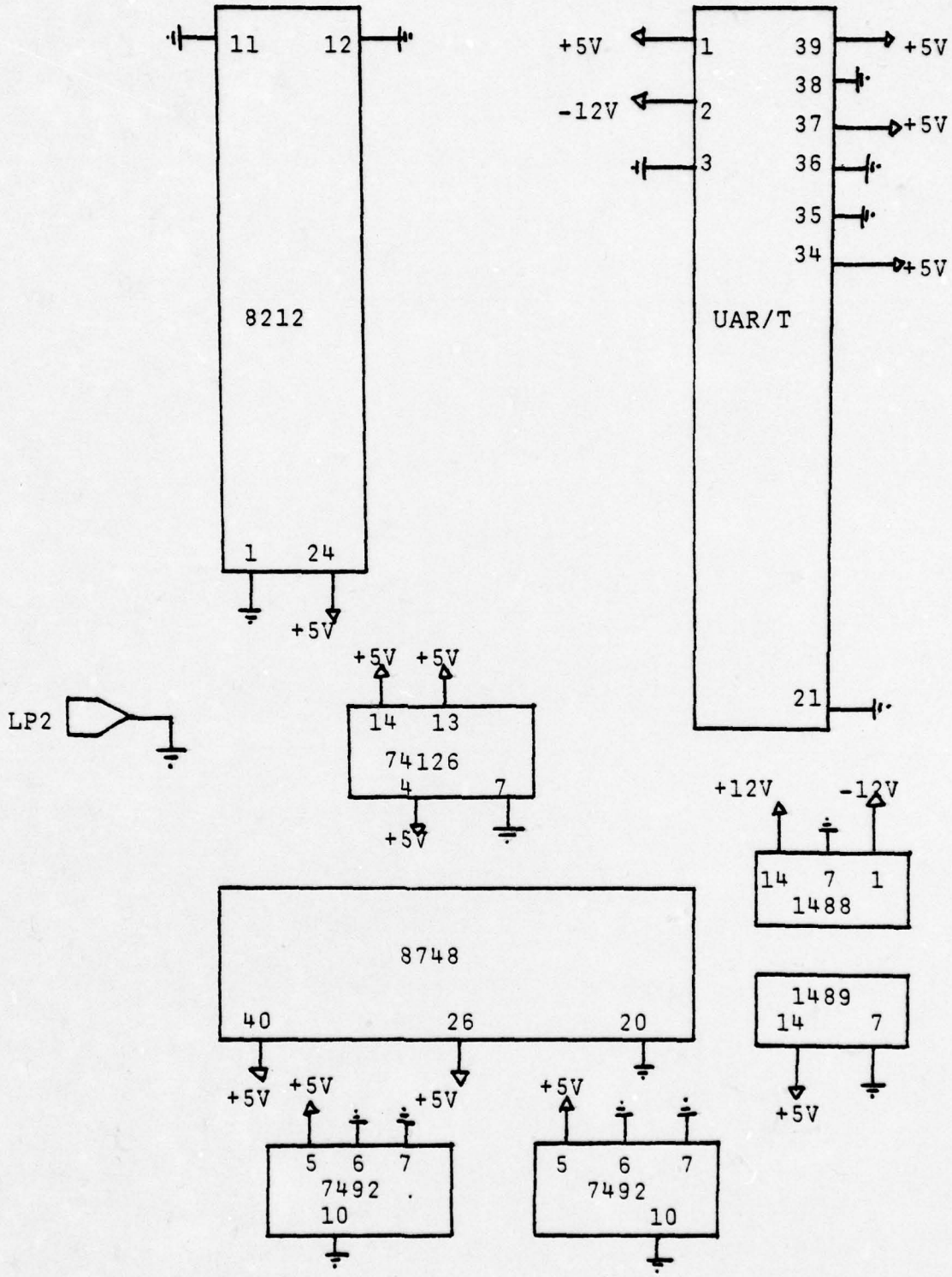


Figure 4 - POWER

The forty pin edge connecting cable from the line printer interface to the expansion interface is identical to the cable from the CPU board to the expansion interface. There are two plastic lugs on each side of the edge connector going to the 32 pin line printer port on the expansion interface. This cable cannot be connected backwards, but it can be connected upside down. The cable should be connected as all the other cables entering the expansion interface.

IV. MICROPROCESSOR DEVELOPMENT AID

The major contribution of this thesis is a disk operating microprocessor development aid for the Z-80 microprocessor. Since the INTEL 8080 language is a subset of the Z-80 language, this development aid is, in all respects, an 8080 microprocessor development aid as well. Since the Naval Postgraduate School has other and more sophisticated development aids for the 8080, the use of this system will find its most pratical use with the Z-80.

The cost of the TRS-80, as presently configured, compares favorably with the prompt-80/85. The system of basic language programs which operates the TRS-80 as a microprocessor development aid performs all of the functions of the prompt-80/85 with the exception of the PROM operations. Consideration will be given to extending this Z-80 development aid to include PROM operations. In addition to the prompt-80/85 functions, this system has a single step execution operation which allows either hardcopy or video display of the registers after each step.

A. THE SYSTEM

The entire basic program to support the microprocessor development aid takes less then 11K of RAM. In a TRS-80 with 16K of RAM that would leave about 5K of RAM for the machine language programs. To facilitate the use of the system and memory usage, the TRS-80 disk operating system (TRSDOS) is used.

The software to support the TRSDOS uses about 10K of RAM, leaving less than 6K of RAM for the system programs. The basic programs are segmented into functional groups using less than 2K of RAM each. Each of these small programs which are resident on the disk are called from the disk by the monitor program. Each of the program segments calls another segment as its last step, thus the system of programs loop from one to another.

With 10K of RAM used by TRSDOS and about 2K of RAM used by the basic programs, this leaves a little more than 3K of RAM for the assembled assembly language programs. Any time an assembly language program of greater than 3K is being developed, the developer should think about a higher level language to produce his software. When the Naval Postgraduate School's TRS-80 RAM is expanded to 48K, the same restriction should be followed.

On system start up, the number of files question should be answered by pressing ENTER; this defaults with three files. Since each file requires about 285 bytes of RAM, any answer other than the default will rob the system of the RAM required for executing its program. The memory size question should be answered with 29630 ENTER. This protects 3K of RAM for the user's assembly language programs.

The use of this system is very much dependent upon a strong background in 8080 and Z-80 assembly language. References 7, 8, 9, and 10 are recommended for the beginner in that order. An understanding of the assembly language and how it operates within the microprocessor cannot be replaced by this system of basic language programs.

B. THE BASIC PROGRAMS

The programs used to support the microprocessor development aid are written in TRS-80 Disk Basic, reference 5 and 6 apply. Their execution is slower than that of similar operating systems which are assembly language based. An asserted attempt has been made to have the inputs and the outputs closely resemble their counterparts in T-BUG and the TEKTRONIX 8002 systems. The programs are interactive in nature and require a developer's response at appropriate times during execution.

Examples of the use of these programs are given as sample labs in Appendices X, Y, and Z. Use of these labs will be discussed later in this chapter.

All of the basic programs, but one, are protective files as defined in Ref. 6. They cannot be loaded, altered or executed without the use of their password. The password used by all the programs is DFC, the initals of the author. The disk's master password is the same as given in Ref. 6. It is not necessary to use or even know either password when using this microprocessor development aid.

The basic programs of Appendices C through N, and P through S are identical in content to those resident on the disk. However, to save memory locations, unnecessary spaces have been deleted for the programs on the disk. Since this entire thesis is being generated by the TPS program, on the IBM 360, the appendices have additional spaces generated by TPS to make them more pleasing to the eye. Additionally the subroutine calls, which appear in most of the programs, call line numbers not present in Appendices C through N, and P through S. These subroutines appear in Appendices T, U, and V, but are resident on the disk in the basic program when required.

1. Monitor Program

The monitor program, Appendix C, functions as its name implies. It establishes a starting position for the microprocessor development aid and allows the operator to chose the type of operation which he wants to accomplish. To start the monitor and, therefore, start the microprocessor development aid, type RUN"S" and then press ENTER. This will load and run the monitor program. "S" is a program which is one line long; that line is RUN "MONITOR/BAS.DFC:0" which could be entered directly. What occurs on RUN "S" is that the monitor program is found on disk drive 0 and loaded into the RAM and then executed.

The first requested input from the monitor program is a request for one of two letters S or L. These stand for a short (very short) or long explanation of the commands which are recognized by the monitor program. If the operator enters some character, group of characters, or the null character (by just pressing ENTER), the program defaults by responding with the short explanation.

After the operator responds to and receives his explanation, the program asks for a mode of operation. The operator must then respond with one of a set of characters used in the explanation. If the operator enters a different string of characters, then what appears in the explanation (spaces count), the program defaults with the short explanation. It will continue giving the short explanation until a proper string of characters is entered.

Upon receiving a proper string of characters the monitor program directs the locating, loading and execution of a new program. At the end of this, the monitor program is

no longer in the RAM. The monitor program is reloaded as a final step to each of the other modes of operation. Thus, the microprocessor development aid is self perpetuating.

If, for any reason, the operator requires to get to the monitor program at some other time, press BREAK then type RUN "S" and press ENTER.

2. Dump Mode

To start the dump mode program, Appendix D, from the monitor program type DUMP, then press ENTER when the mode is requested. This program will provide, either in hard copy or on the video display, the contents (in hexadecimal code) of any memory location or string of memory locations up to and including the entire 64K of RAM addressable by the Z-80.

The operator must provide, when asked, a starting memory location in decimal, the number of memory locations in decimal and an indication of video or line printer output. There is no default on the type of output, so if the operator enters something other than a single letter V for video or the two letter string LP for the line printer, there will be no output. The data is outputted in the following format per line: starting memory location of the line in decimal; starting memory location of the line in hexadecimal; followed with sixteen bytes of data from the next sixteen memory locations.

After the requested data is outputted, the program gives the operator an opportunity to start the dump program again. The program defaults on any character string other than YES to the monitor program.

3. Load Mode

The load mode program, Appendix E, is entered by typing LOAD and pressing ENTER when the mode is requested during the monitor program. This program operates almost identically to the "exam" command of the TEKTRONIX 8002. The operator will be required to enter a starting memory location in decimal. The location should be, but is not limited to, the reserved RAM locations 7400 to 7FFF (29696 to 32767 decimal) which are protected on startup. Since this program changes the contents of RAM locations, to use it at RAM location other than 7400 to 7FFF Hex invites disaster, for the operator can end up changing something that should not be changed.

The program outputs the hexadecimal memory location and its current contents. When a question mark (?) appears, the operator can change the memory location by typing a two digit hexadecimal code and pressing ENTER. The program defaults using only the first two characters of any multi-character string. If no characters are typed and ENTER is pressed, the memory location is unchanged and the program proceeds to the next memory location. The memory location always increases and never decreases.

To stop the loading process type QUIT and press ENTER. This will start the monitor program.

4. Execution Mode

The execution mode program, Appendix F, is entered by typing EXEC and pressing ENTER when the mode request is made during the monitor program. This program requires a decimal number which is used as a starting memory location.

The program transfers control so that the machine language code is executed starting at that memory location. This execution is similar to the GO command of the prompt 80/85. The execution is stopped and control transferred back to the basic language program upon execution of a return which was not proceeded by a call in the assembly language program. The USR0 instruction is used to transfer control. See Refs. 5 and 6.

When execution is complete, the operator is asked if further execution is required. If so, a YES is typed and ENTER pressed, then the basic language execution mode program is restarted asking for a starting memory location in decimal. The program defaults on any other character string than YES to the monitor program.

The use of this mode should be limited to assembly language programs which have already been debugged. Since the execution is very rapid, at the Z-80 clock rate, any attempt to see single steps would be useless. Techniques for writing Z-80 assembly language programs for this system are discussed later.

5. Single Step Execution Mode

The single step execution mode is a system of eight basic language programs, Appendices G through N, and one machine language program, Appendix O. The system of programs is started by typing SS EXEC and pressing ENTER in response to the mode question of the monitor program. Single step execution will provide, either in hardcopy or on the video display, the program counter of the step being executed. The contents of each working register and the stack pointer, after the step is executed, is also outputted. The primary use of this set of programs is as a

debugging aid. For long machine language programs the hardcopy output is essential, since the video display will contain no more than twelve lines of output. What follows is a functional explanation for each of the nine separate programs which make up the single step execution mode.

Single step execution program, Appendix G, is the executive program of the single step execution mode. It is run only once during the single step procedure, and this does the preliminary accounting procedures required for execution. It first loads the machine language program, appendix O, into memory. The question of output is then addressed. The operator responds with a V or LP for either video or line printer output. The type of output is stored in the least significant bit of memory location 73D0 Hex. There is no default for output; a zero in this location causes video output and a one causes line printer output. If some other string of characters is used to answer the output question, the contents of this bit remains unchanged and the output is either video or line printer. The last entry by the operator in this mode is a starting memory location in decimal. This location is now used by the system of programs as the program counter and is stored in memory location 73D2 and 73D3. Since the single step mode is using several programs, any information to be passed between the programs must be stored in the protected part of the RAM. Appendix W shows a memory map of the protected portion of the RAM used by the single step execution mode.

The Appendix G program, as a final step, loads and executes single step execution program 1, Appendix H. This, as well as the remaining single step execution programs, has no interactions with the operator. The program sets memory location 73D1 to zero. Memory location 73D1 is called K throughout this set of programs. The output bit from memory location 73D0 is checked and a header is outputted as

required. This header labels each of the columns of the output. The single step execution program 1 is returned to, after ten machine language program steps are accomplished. This process would be accomplished as a do-loop if the single step execution mode was one, not eight basic programs.

The Appendix H program, as a final step, loads and executes single step execution program 2, Appendix I. This program performs a very fundamental function of a single step routine; it translates the hexadecimal code of the contents of the program counter's memory location into the length in bytes of the instruction to be executed. The variable N2 is used to store the number of bytes per current instruction, see Appendix W. Instead of using this variable in single step execution program 2, the program loads, then executes one of the four programs, Appendices J through M, depending on the number of bytes.

If single step execution program 3 is executed next, then the instruction to be executed is one byte in length. There are four distinct classes of one byte instructions in the Z-80 language. There is a class which does not change the program counter except to move it down to the next instruction. Ninty percent of the single byte instructions are like this. Another class is one that only changes the program counter. The only single byte instruction of this class is the JP(HL) instruction. The third class may change the program counter and the stack pointer. These are the return and conditional return instructions. The last class are the restart instructions. The single step execution program 3 separates these four classes of single byte instructions and set control codes to insure their proper execution within the single step framework. An instruction which does not change the program counter will set M5 to 1 and M6 to 3. An instruction to jump to HL will set M5 to 1,

M6 to 1, and places the new program counter in memory locations 73E8 and 73E9. The returns check to see if the return has had a corresponding call. The conditional returns are checked for the truth of the condition. If the return has a corresponding call, the returning program counter is placed in memory locations 73E8 and 73E9; the stack pointer is caused to increment twice; M5 is set to 1 and M6 is set to 1. If the return is a conditional return failing its condition, M5 is set to 1 and M6 is set to 2. If the return does not have a corresponding call, then it is an end of program return; M5 is set to 2 and M6 is set to 1. A restart instruction is not executed since its execution would transfer control to a Level II basic ROM memory location. A restart sets M5 to 1, M6 to 2, and prints a diagnostic on the video display. M5 and M6 are control variables which are stored in memory location 73D0, prior to the loading and execution of single step execution program 7, Appendix N, as a last step of this program.

Single step execution 4, Appendix K, is executed next, if the instruction being executed is two bytes long. There are four classes of the two byte instructions. One type is executed directly and does not change the program counter. These instructions set M5 to 1 and M6 to 3. Another class are the jumps. The JP(IX) and JP(IY) instructions each set M5 to 1 and M6 to 1, then place a new program counter in memory locations 73E8 and 73E9. The relative jump instruction decodes the new program counter and then does the same as JP(IX). The conditional relative jumps with a true condition act as a relative jump. If a false condition exists, then M5 is set to 1 and M6 is set to 2. If the special decrement relative jump (DJNZe) is being executed, then, in addition to the conditional relative jump procedure, the B register must be decremented. The third class is the return from interrupt and return from nonmaskable interrupt. This class sets M5 to 1 and M6 to 2,

and then outputs a diagnostic to the video display. The instructions RETI and RETN are emulated with no operations (NOP). The last class of two bytes instructions are the iterative load, compare, input and output instructions (LDIR, LDDR, CPIR, CPDR, INIR, INDR, OTIR, and OTDR). They set M5 to 1 and M6 to 3 which allows their execution to completion. A diagnosic is outputted to the video which explains that these instructions are executed directly with only one line of output. The control variables M5 and M6 are then stored in memory location 73D0; and the single step program 7, Appendix N, is loaded and executed.

Single step execution program 5, Appendix L, is executed if the instruction being executed is three bytes in length. There are three classes of three byte instructions. The ones which do not change the program counter are executed directly. The control variables M5 is set to 1 and M6 is set to 3. The second class is the jump and conditional jumps. The jump will set M5 to 1 and M6 to 1, and store the new program counter in memory locations 73E8 and 73E9. A conditional jump is checked for the truth of the condition. If the condition is true, then it is treated like a jump. If the condition is false, then M5 is set to 1 and M6 is set to 2. The third class is the call and conditional calls. These act like jumps and conditional jumps with additional actions. The current program plus three must appear to be pushed on the stack; the stack pointer decreased by two; and M4 which is stored at 73D4 is decreased by one. The control variables M5 and M6 are stored at 73D0; then single step execution program 7, Appendix N, is loaded and executed.

Single step execution program 6, Appendix M, is executed if the instruction being executed is four bytes long. The Z-80 language does not have any four byte instruction which changes the program counter. This program

sets the control variables M5 to 1 and M6 to 3, and then stores them in memory location 73D0. The single step execution program 7, Appendix N, is then loaded and executed.

Single step execution program 7, Appendix N, is the ending executive program for each machine language program step. The program decodes K which is the number of lines that this program has printed since the last register header of single step execution program 2 was printed. LP is decoded. If LP is one, then output goes to the line printer. If LP is zero, then output goes to the video display. N3 is decoded as the current program counter. If a jump, call or return is to be executed, then N1 is decoded as the new program counter. M5 is decoded as 1 or 2. M6 is decoded as 1, 2 or 3. N2 is decoded as the byte length of the machine language instruction being executed. If M6 is 1, then the new program counter, N1, is stored in 73D2 and 73D3 and the machine language program of appendix O is executed. If M6 is 2, then N1 is set to N2 plus N3 (i.e., the new program counter is the old program counter plus the number of bytes of the executing instruction); and the steps done for M6 equal to 1 are carried out. For both M6 set to 1 or 2, the contents of 73E8 to 73EB are either no operations (NOP) or previously set in single step execution programs 3, 4 or 5. If M6 is 3, then the instruction being executed is placed in 73E8 to 73EB (whith NOP's in empty locations); and the steps used when M5 is 2 are carried out. Upon returning from the machine language program of appendix O, the program counter, registers and stack pointer are assembled and outputted as the variable LP dictates. If M5 is 2, then the machine language program is over (one more return than calls). An end of execution message is printed on the video display and the monitor program is loaded and executed. If M5 is 1, then one is added to K. If K is greater than 9 (i.e. ten lines of output has been

sets the control variables M5 to 1 and M6 to 3, and then stores them in memory location 73D0. The single step execution program 7, Appendix N, is then loaded and executed.

Single step execution program 7, Appendix N, is the ending executive program for each machine language program step. The program decodes K which is the number of lines that this program has printed since the last register header of single step execution program 2 was printed. LP is decoded. If LP is one, then output goes to the line printer. If LP is zero, then output goes to the video display. N3 is decoded as the current program counter. If a jump, call or return is to be executed, then N1 is decoded as the new program counter. M5 is decoded as 1 or 2. M6 is decoded as 1, 2 or 3. N2 is decoded as the byte length of the machine language instruction being executed. If M6 is 1, then the new program counter, N1, is stored in 73D2 and 73D3 and the machine language program of appendix O is executed. If M6 is 2, then N1 is set to N2 plus N3 (i.e., the new program counter is the old program counter plus the number of bytes of the executing instruction); and the steps done for M6 equal to 1 are carried out. For both M6 set to 1 or 2, the contents of 73E8 to 73EB are either no operations (NOP) or previously set in single step execution programs 3, 4 or 5. If M6 is 3, then the instruction being executed is placed in 73E8 to 73EB (whith NOP's in empty locations); and the steps used when M5 is 2 are carried out. Upon returning from the machine language program of appendix O, the program counter, registers and stack pointer are assembled and outputted as the variable LP dictates. If M5 is 2, then the machine language program is over (one more return than calls). An end of execution message is printed on the video display and the monitor program is loaded and executed. If M5 is 1, then one is added to K. If K is greater than 9 (i.e. ten lines of output has been

generated), then the program loops to single step execution program 1. If K is less than or equal to 9, then the program loops to single step execution program 2.

Figure 5 shows a flow chart of the single step execution programs. The nested operation of these programs takes twenty seconds of real time to execute one step. This gives the operator plenty of time to anticipate the changes which each step will make to the registers. For an experienced operator, it is far too long to wait for the next step. For that case using a hardcopy output is suggested.

6. Disk Storage Mode

To start the disk storage mode program, Appendix P, from the monitor program type DSTORE, then press ENTER when the mode is requested. This program will store a machine language program on the disk. Even more generally, it will store any portion of memory on any disk, if more than one disk is connected to the system.

The operator must provide a starting memory address in decimal, an ending memory address in decimal, and a filename. Reference 6 is the source for information on the optional parameters for filenames. The simplest filename is a name of no more than eight alpha-numeric characters. The data is stored under the filename, therefore an easily remembered filename should be used. The program stores the data, reminds the operator of the filename in use, and then loads and executes the monitor program.

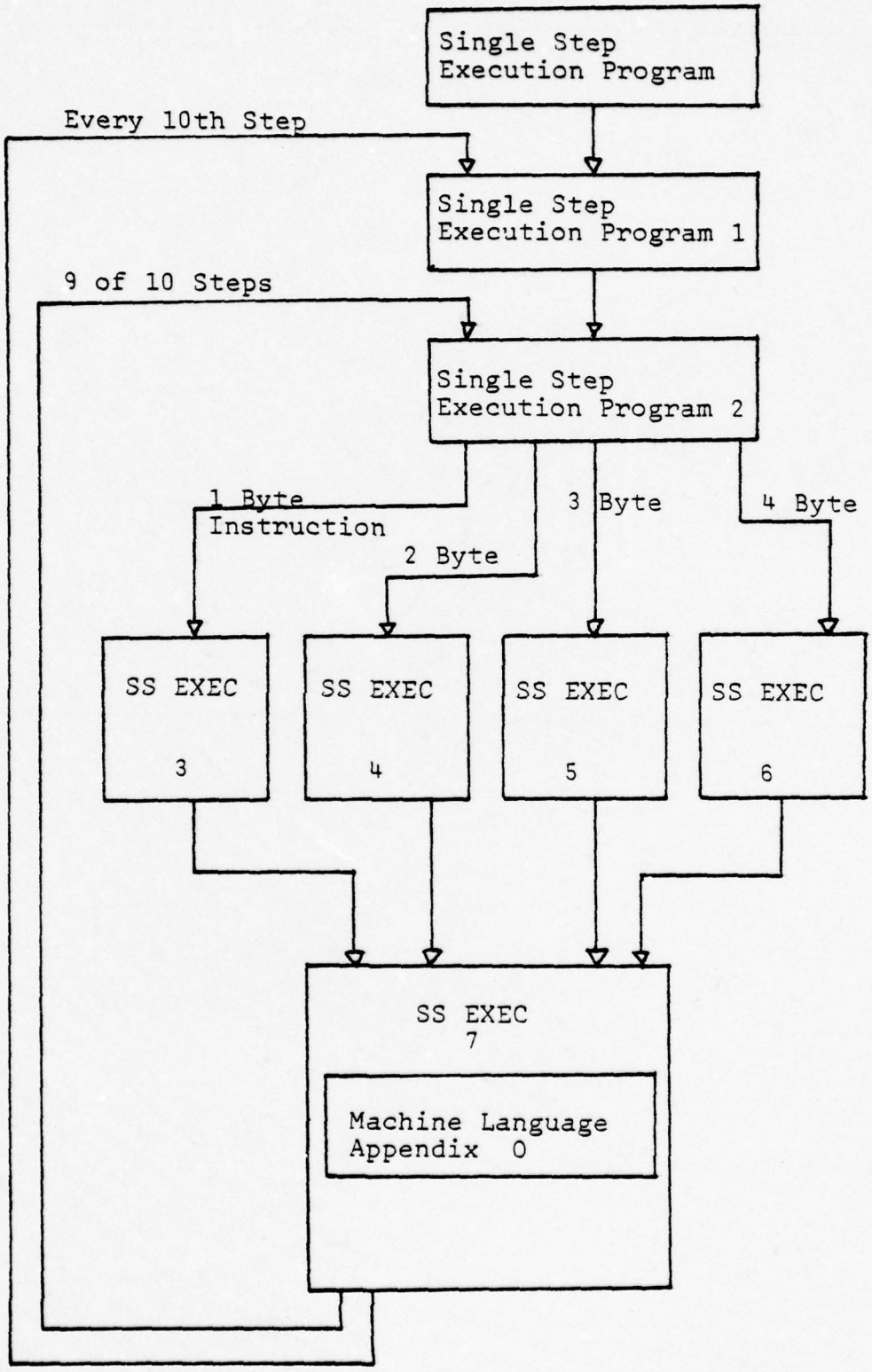


Figure 5 - SINGLE STEP EXECUTION FLOW CHART

7. Disk Recall Mode

To start the disk recall mode program, Appendix Q, from the monitor program type DRECALL, then press ENTER when the mode is requested. This program will load a machine language program into memory from a data file on the disk created by the DSTORE program.

The operator must provide a starting memory address in decimal, an ending memory address in decimal, and the filename of the data file used during DSTORE to create the file. After the machine language program is loaded, the operator is reminded which memory locations have been loaded. The monitor program is now loaded and executed.

8. Tape Storage Mode

To start the tape storage mode program, Appendix R, from the monitor program type TSTORE, then press ENTER when the mode is requested. This program will store a machine language program on the tape recorder.

The operator must provide a starting and ending memory address in decimal. The tape recorder must be on and set to record. There is no filename for the tape recorder storage in this application, so remember the starting and ending tape counter. The data is stored on the tape, sixteen bytes at a time. After the data is transfered to the tape, the monitor program is loaded and executed.

9. Tape Recall Mode

To start the tape recall mode program, Appendix S, from the monitor program type TRECALL, then press ENTER when the mode is requested. This program will load into the RAM memory a machine language program which has been stored on the tape using TSTORE mode.

The operator must provide a starting and ending memory address. The program expects the operator to place the tape recorder in play at about the starting tape counter used in the TSTORE mode. After the data is loaded in the RAM, the monitor program is loaded and executed.

C. SPECIAL CONSIDERATIONS FOR MACHINE LANGUAGE PROGRAMMING

Most systems, which support machine language programming, have peculiarities of which a programmer can take advantage. As presently configured, this microprocessor development system has several peculiarties. There is only 3K of protected RAM for machine language usage. There is a 1K video RAM which might be utilized. There is an expansion port to which hardware may be added. There is a memory mapped keyboard which can be used. There is the possibility of using the subroutines of the level II basic ROM as macro instructions.

The 3K of protective RAM, memory addresses 7400 to 7FFF, is available for machine language programs, a stack, and, as required, for storage of data. The machine language program resident in this portion of RAM is not relocatable to another section of the RAM. If a large amount of stack space is required, a program's integrity may be in jeopardy. When the RAM is expanded to 48K, none of the current software will support the use of the additional 32K of RAM. The PEEK and POKE instructions require a different syntax in

the supporting basic programs. The PEEK syntax in the first 16K of RAM is PEEK(address). The PEEK syntax for the RAM in the expansion interface, at memory addresses above 32767, is PEEK(-1*address-32676). To PEEK at address 40000 the instruction would be PEEK(-1*7233). this limitation in the support software will not prevent the programmer from using this additional memory as a stack or data storage by a machine language program.

There is 1K of video RAM, memory locations 3C00 to 3FFF, which can be used for several purposes. It is used in the example labs as a indicator light. By loading BFhex into a video address the video location is whited out. This can take the place of a single hardware LED. The ASCII character codes and video codes which are available are given in appendix C of Ref. 5. The video RAM may be used for the stack if required. Use of the video RAM is limited to applications which will not interfere with the TRS-80's use of the video RAM. For example, the programmer would not find the video RAM useful for a stack during a single step execution with output to the video display. However, the stack could use the video RAM during single step execution if output was to the printer. The second example lab does use the video RAM during a single step execution with output to the video display.

The use of the expansion port, called the screen printer port on the expansion interface, will be useful for hardware projects which would otherwise require a dedicated Z-80 CPU. Figure 22 of Ref. 1 is a detailed description of the expansion port edge connecter. In general, each line of this edge connector will drive only one TTL load. Therefore, drivers or tri-state buffers will be required for an external hardware project. The five volt power supply, pin 39, has been modified to show a ground.

The use of the keyboard and other ROM subroutines will add a considerable dimension to machine language programming. At present, Ref. 9 is the sole source for ROM subroutine memory locations. In the future, the Naval Postgraduate School will have a complete listing of ROM subroutines for the programmer to utilize.

D. USING THE DEVELOPMENT SYSTEM

The learning curve is a substantial barrier to the use of any new product or device. The same is true for any Z-80 based system or project or the development system used to create the required software. To have a system of basic language programs which are described as a microprocessor development aid for the Z-80 is not enough. A developer must have confidence in the system and must be helped to get past the largest part of the learning curve. Appendices X, Y, and Z are designed as training sessions to help a new developer to learn the system, to develop confidence in the system's ability to solve his problem, and to start with small successes on which larger projects can be built.

1. Lab 1

Appendix X, lab 1, first contains a very detailed power up procedure. The system has six power cords, three edge connector cables, three connecting cables, and one RS-232 connecting cable. The sequence of powering up is important and should be followed each time by the operator. There are many other capabilities of the TRS-80 microcomputer system. The operator should refer to Refs. 5 and 6 when these other capabilities are required.

The operator is then led through a sequence of operations which will retrieve a machine language program resident on the mini disk and places it in the protective portion of the RAM using the disk recall mode. The contents of the RAM is verified using the dump mode. This program is modified by the load mode and executed by the execute mode. The program uses the video RAM as an output and whitens the first 256 video RAM locations.

The return at the end of the program which exists without a call is a signal to the TRS-80 to exit this machine language program and return to the basic language program. All programs written for this system must use an ending return. This return should be removed when the program is used in a stand alone Z-80 system.

At the end of lab 1, the operator is expected to understand the start up procedure, what is done by the monitor program, and how some of the modes work. He should examine the machine language program at the end of this lab to gain an appreciation for what the program is accomplishing.

2. Lab 2

Lab 2, Appendix Y, builds on the experience of lab 1. A more complicated machine language program is used to demonstrate the use of the single step execution mode. There are four levels of calls during this program. The video memory location 3FF8 is used as an LED and follows the line of registers, which produced it, up the video display as the video display scrolls. When the machine language program of lab 2 is single step executed using line printer output, the video memory location is lit, but not scrolled.

The operator is to obtain an appreciation for the debugging potential of the single step execution mode of operation. He should also notice the absence of the prime registers from the single step execution printout. If the programmer requires the use of both sets of Z-80 registers, then he must change the program for it will not execute properly using the single step execution mode.

3. Lab 3

Lab 3, Appendix Z, is a structured example of the system's use. The operator should begin this lab with a machine language program to debug. If the operator fails to write his own program, then a short program can be found at the end of the lab for this purpose. At the end of this lab the operator will have operated all of the microprocessor development system. He has passed the largest portion of the system's training curve and is ready to use the system for his own software development.

V. CONCLUSIONS AND RECOMMENDATIONS

What was demonstrated with the preceeding is that an inexpensive microcomputer system can be programmed to perform many of the functions of a dedicated microprocessor development system costing up to ten times as much. Speed of operation was sacrificed for the ease of programming. The use of Radio Shack's disk basic language is more than adequate to support this Z-80 software development system. The housekeeping operations, which are common to development systems, are general enough to support any other microprocessor development system.

A. FUTURE SOFTWARE

The software future of the TRS-80 microcomputer system is limited only by the ideas implemented with it. To support the Z-80 Radio Shack has distributed two software packages. T-BUG, Ref. 9, is a machine language program that will load, store on tape, dump, and execute with or without breakpoints. The TRS-80 Editor/Assembler, Ref. 10, is a machine language program that will edit and assemble assembly language Z-80 programs. These machine language programs both take advantage of the basic ROM's subroutines. A major disadvantage of these programs is that they cannot be used with the TRS-80 disk basic language. To improve this Z-80 microcompiter development aid, a basic language program to relocate the T-BUG and Editor/Assembler machine language programs to a portion of memory which is not used by disk basic is necessary to allow the use of these

programs with disk basic.

A list of basic ROM subroutines which could be used as macro instructions in other machine language programs would add a new dimension to programming. To obtain these subroutines will require a disassembler program and a lot of bookkeeping.

To expand the TEKTRONIX 8002 to include additional microprocessors will cost about two thousand dollars per additional microprocessor. The software required for the TRS-80 to support additional microprocessors will have little direct cost. To expand the TRS-80 to include development systems for microprocessors already supported by the TEKTRONIX 8002 is not a time effective proposal. The software for the TRS-80 should be expanded to include INTEL's 8748 microprocessor family. The 8748 microprocessor has the greatest potential for class project work. A development system for this family of microprocessors will go a long way to ease the burden of software development for the 8748.

B. FUTURE HARDWARE

There are several cost effective hardware additions to the TRS-80 microcomputer which will support a microprocessor development system. An interface between the TRS-80 and the TEKTRONIX 8002 will provide an expanded disk capability, the sharing of an additional 64K of RAM, and the use of a PROM programmer. An interface between the TRS-80 and the prompt-48 will provide additional hardware for a 8748 microprocessor development system. The use of the prompt-48's PROM programmer and execution with or without breakpoints will relieve some of the hardware requirements

for such a system. An interface between the TRS-80 and the IBM 360 will add mass storage capability to any system.

There are several new PROMs and at least one old PROM which are not supported by the equipment available in the microcomputer lab. The TRS-80 with its basic language capability would be a logical choice of systems to expand to support these PROMs. A class project could be directed toward one PROM or a family of PROMs.

APPENDIX A

THE PROCEDURE FOR THE CONVERSION TO LEVEL II BASIC FROM LEVEL I BASIC

1. Open CPU/keyboard cabinet.
2. Remove Level I basic ROM's from sockets Z33 and/or Z34.
3. Plug in the 24 pin header of the ROM board to Z33 or Z34 (Z33 is recommended).
4. Attach the ROM board to the etched side of the CPU board.
5. Replace DIP shut X3 with the new DIP shut.
6. Check the color code: yellow (A11), red (A12), orange (A13), green (ROM*).
   a. Solder the yellow wire to pin 5 of Z37.
   b. Solder the red wire to pin 13 of Z21.
   c. Solder the orange wire to pin 3 of Z21.
   d. Solder the green wire to pin 8 of X3.
7. Close CPU/keyboard cabinet.

APPENDIX B

THE 8748 PROGRAM TO GENERATE A LINE FEED

MEM LOC	HEX CODE	MNEMONIC	COMMENTS
000	04	JMP	Power Up Restart
001	05		
002	00	NOP	
003	04	JMP	Interrupt Vector
004	28		
005	B8	MOV R0,44	
006	44		
007	B9	MOV R1,7F	
008	7F		
009	BA	MOV R2,0D	
00A	0D		
00B	BB	MOV R3,0A	
00C	0A		
00D	00	NOP	
00E	00	NOP	
00F	00	NOP	

MEM LOC	HEX CODE	MNEMONIC	COMMENTS
010	23	MOV A,01	
011	01		
012	3A	OUTL P2,A	Hi-Z on DS(not) from 8748
013	00	NOP	
014	23	MOV A,FF	
015	FF		
016	39	OUTL P1,A	Enables P1 for OR's and AND's (not necessary in this Application)
017	00	NOP	
018	81	MOVX A,R1	Places BUS in Hi-Z State
019	00	NOP	
01A	75	ENTO CLK	Makes T0 a clock at 1/3 The 8748 Clock
01B	05	EN I	Enable Interrupt
01C	00	NOP	
01D	00	NOP	
01E	00	NOP	
01F	00	NOP	

MEM LOC	HEX CODE	MNEMONIC	COMMENTS
020	00	NOP	
021	F8	MOV A,R0	

022	39	OUTL P1,A	Enable 8212 and Enable TRS-80
023	04	JMP	Loop to 020 Until Interrupted
024	20		
025	00	NOP	
026	00	NOP	
027	00	NOP	
028	23	MOV A,66	An Interrupt has Occurred
029	66		
02A	39	OUTL P1,A	Stop TRS-80
02B	08	INS A,BUS	Input Last Character
02C	59	ANL A,R1	Insure Character is 7 Bits
02D	DA	XRL A,R2	Check Last Character for Carriage Return
02E	00	NOP	
02F	00	NOP	

MEM LOC	HEX CODE	MNEMONIC	COMMENTS
030	96	JNZ	If No Carriage Return Start End Interrupt Sequence
031	41		
032	00	NOP	
033	86	JNI	Loop to 032 Until EOC=1

```
                                        From UART
034         32
035         00          NOP
036         23          MOV A,33
037         33
038         39          OUTL P1,A   Disable 8212
039         FB          MOV A,R3
03A         02          OUTL BUS,A  Place 0A on BUS for
                                    A Line Feed
03B         23          MOV A,00
03C         00
03D         3A          OUTL P2,A   P2 is Low to Start
                                    The Strobe to UART
03E         23          MOV A,11
03F         11

MEM LOC   HEX CODE   MNEMONIC    COMMENTS
  040       3A       OUTL P2,A   End of Strobe to UART
  041       81       MOVX A,R1   Place BUS in Hi-Z State
  042       00       NOP
  043       86       JNI         Loop to 042 Until EOC=1
  044       42
  045       00       NOP
  046       46       JT1         Jump to 04A When Printer
```

```
                                        Is Ready

047         4A

048         04          JMP             Loop to 045 Until Printer

                                        Is Ready

049         45

04A         93          RETR            Return from Interrupt
```

APPENDIX C

MONITOR PROGRAM

```
500 ' MONITOR: A Z-80 MICROPROCESSOR DEVELOPMENT AID FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79
501 DEFSTR A-B
502 PRINT "MONITOR PROGRAM"
504 PRINT "A Z-80 MICROPROCESSOR DEVELOPMENT AID"
506 INPUT "SHORT OR LONG EXPLANATION? INPUT S OR L";A
508 IF A="S" PRINT "DUMP,LOAD,EXEC,SS EXEC,TSTORE,TRECALL, DSTORE,DRECALL,END,L"
510 IF A="S" INPUT "MODE";B
512 IF A="S" GOTO 534
514 PRINT "INPUT: DUMP, TO EXAMINE MEMORY LOCATIONS"
516 PRINT "INPUT: LOAD, TO ENTER PROGRAM INTO MEMORY"
518 PRINT "INPUT: EXEC, TO RUN A PROGRAM"
520 PRINT "INPUT: SS EXEC, TO SINGLE STEP THROUGH A PROGRAM"
522 PRINT "INPUT: DSTORE, TO STORE A PROGRAM ON THE DISK"
524 PRINT "INPUT: DRECALL, TO RETRIEVE A PROGRAM STORED ON THE DISK"
526 PRINT "INPUT: TSTORE, TO STORE A PROGRAM ON THE TAPE RECORDER"
528 PRINT "INPUT: TRECALL, TO RETRIEVE A PROGRAM STORED ON
```

```
THE TAPE RECORDED"
530 PRINT "INPUT: END, TO EXIT THIS DEVELOPMENT AID"
532 INPUT B
534 IF B="DUMP" RUN "DUMP/BAS.DFC:0"
536 IF B="LOAD" RUN "LOAD/BAS.DFC:0"
538 IF B="EXEC" RUN "EXEC/BAS.DFC:0"
540 IF B="SS EXEC" RUN "SSEXEC/BAS.DFC:0"
542 IF B="DSTORE" RUN "DSTORE/BAS.DFC:0"
544 IF B="DRECALL" RUN "DRECALL/BAS.DFC:0"
546 IF B="TSTORE" RUN "TSTORE/BAS.DFC:0"
548 IF B="TRECALL" RUN "TRECALL/BAS.DFC:0"
550 IF B="L" GOTO 514
552 IF B="END" END
554 A="S":GOTO 508
```

APPENDIX D

DUMP PROGRAM

```
5 'DUMP: A PART OF A Z-80 MICROPROCESSOR DEVELOPMENT AID FOR
DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79
10 CLEAR 300:DEFSTR A-B:DEFINT I-N
15 PRINT "DUMP MODE"
20 INPUT "INPUT STARTING MEMORY LOCATION IN DECIMAL";N1
25 INPUT "INPUT NUMBER OF MEMORY LOCATION IN DECIMAL";N2
30 PRINT "IS OUTPUT VIDEO OR FOR THE LINE PRINTER?"
35 INPUT "INPUT V OR LP";A
40 N3=N1
44 B1=STR$(N1)
45 N=N1:O=4:GOSUB 100:B1=B1+" "+B
50 FOR K=1 TO 16
55 N=PEEK(N1):O=2:GOSUB 100:B1=B1+" "+B:N1=N1+1
65 NEXT K
70 IF A="V" PRINT B1
75 IF A="LP" LPRINT B1
80 IF N1<N3+N2 GOTO 44
85 INPUT"CONTINUE DUMPING (YES OR NO)";B
90 IF B="YES" THEN 20 ELSE RUN "MONITOR/BAS.DFC:0"
```

APPENDIX E

LOAD PROGRAM

```
5 'LOAD: A PART OF A Z-80 MICROPROCESSOR DEVELOPMENT AID FOR
DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

6 GOTO 300

300 DEFINT I-O:DEFSTR A-B

305 PRINT "LOAD MODE"

310 INPUT "INPUT STARTING MEMORY LOCATION IN DECIMAL";N1

315 PRINT "OUTPUT IS VIDEO - USE QUIT TO STOP LOADING"

320 N=N1:O=4:GOSUB 100:B1=B

325 N=PEEK(N1):O=2:GOSUB 100:B2=B

330 B3=B1+" "+B2:PRINT B3:INPUT B

335 IF B="QUIT" GOTO 350

336 IF B="" GOTO 345

340 GOSUB 200:POKE N1,N

345 N1=N1+1:GOTO 320

350 RUN "MONITOR/BAS.DFC:0"
```

APPENDIX F

EXECUTION PROGRAM

```
6 GOTO 600

600 'EXECUTION: A PART OF A Z-80 MICROPROCESSOR  DEVELOPMENT
AID FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

605 DEFSTR A-B:DEFINT I-O

610 PRINT "EXECUTION MODE"

615 INPUT "INPUT STARTING MEMORY LOCATION IN DECIMAL";N1

620 N=N1:O=4:GOSUB 100:PRINT "EXECUTION BEGINS AT ",B

625 DEFUSR0=N1

630 PRINT USR0(0)," END EXECUTION"

635 INPUT "CONTINUE EXECUTION MODE (YES OR NO)";B

640 IF B="YES" THEN 610

645 RUN "MONITOR/BAS.DFC:0"
```

APPENDIX G

SINGLE STEP EXECUTION PROGRAM

```
1300 'SS  EXEC: A PART OF A Z-80 MICROPROCESSOR DEVELOPMENT
AID FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

1305 DEFINT I-N:DIM I(19):DEFSTR A-B

1310 I(0)= HED: I(1)= H73: I(2)= HC0: I(3)= H73:  I(4)= H31:
I(5)= HC2:   I(6)= H73:   I(7)= HC1:  I(8)= HD1:  I(9)= HE1:
I(10)= HF1: I(11)= HDD: I(12)= HE1: I(13)= HFD:  I(14)= HE1:
I(15)= HED: I(16)= H7B: I(17)= HCE: I(18)= H73:N=0

1315 FOR N2=29653 TO 29671

1320 POKE N2,I(N):N=N+1

1325 NEXT N2

1330  I(2)= HCE: I(5)= HCE: I(7)= HFD: I(8)= HE5: I(9)= HDD:
I(10)= HE5: I(11)= HF5: I(12)= HE5: I(13)= HD5:  I(14)= HC5:
I(17)= HC0: I(19)= HC9: N=0

1335 FOR N2=29676 TO 29695

1340 POKE N2,I(N):N=N+1

1345 NEXT N2

1350 PRINT "IS OUTPUT FOR VIDEO OR LINE PRINTER"

1355 INPUT "INPUT V OR LP";A

1360 IF A="V" POKE 29648,0

1365 IF A="LP" POKE 29648,1
```

```
1370 INPUT "STARTING MEMORY LOCATION IN DECIMAL";N
1375 N1=N/256:N=N-N1*256
1380 POKE 29650,N:POKE  29651,N1 :POKE  29652,127
1385 RUN "SSEXEC1/BAS.DFC:0"
```

APPENDIX H

SINGLE STEP EXECUTION PROGRAM 1

```
1 'SSEXEC1/BAS.DFC:0
1400 DEFSTR A-B:DEFINT I-Q
1405 POKE 29649,0
1410 K1=(PEEK( H73D0) AND  H01)
1415 A="PC   F   A   B   C   D   E   H   L   SP    IX    IY"
1420 IF K1=0 PRINT A
1425 IF K1=1 LPRINT A
1430 RUN "SSEXEC2/BAS.DFC:0"
```

APPENDIX I

SINGLE STEP EXECUTION PROGRAM 2

```
1 'SSEXEC2/BAS.DFC:0
2 DEFSTR A-B:DEFINT I-Q
6 FOR K1= 29672 TO 29675
7 POKE K1,0
8 NEXT K1
9 N1=256*PEEK(29651)+PEEK(29650):GOTO 27
27 M=PEEK(N1):N=M/16:GOSUB 10:B1=B
28 N=M-N*16:GOSUB 10:B2=B
29  IF  (B1="4"  OR  B1="5" OR B1="6" OR B1="7" OR B1="8" OR
B1="9" OR B1="A" OR B1="B") GOTO 70
30 IF B2="6" OR B2="E" GOTO 71
31 IF B2="1" AND B1=0" GOTO 72
32 IF B1="0" GOTO 70
33 IF B1="1" AND B2="0" GOTO 71
34 IF B1="1" AND B2="1" GOTO 72
35 IF B1="1" AND B2="8" GOTO 71
36 IF B1="1" GOTO 70
37 IF B1="2" AND (B2="1" OR B2="2") GOTO 72
38 IF B2="2" GOTO 72
```

```
39 IF B1="2" AND B2="8" GOTO 71
40 IF B2="A" GOTO 72
41 IF B1="2" AND B2="0" GOTO 71
42 IF B1="2" GOTO 70
43 IF B1="3" AND B2="0" GOTO 71
44 IF B1="3" AND B2="1" GOTO 72
45 IF B1="3" AND B2="8" GOTO 71
46 IF B1="3" GOTO 70
47 IF B2="4" OR B2="C" GOTO 72
48 IF B1="C" AND B2="3" GOTO 72
49 IF B1="C" AND B2="B" GOTO 71
50 IF B1="C" AND B2="D" GOTO 72
51 IF B1="D" AND B2="3" GOTO 71
52 IF B1="D" AND B2="B" GOTO 71
53 IF B2="D" THEN 54 ELSE 70
54 M=PEEK(N1+1):N=M/16:GOSUB 10:B3=B
55 N=M*16:GOSUB 10:B4=B
56 IF B1="D" GOTO 59
57 IF B1="E" GOTO 66
58 IF B1="F" THEN 59 ELSE STOP
59 IF B3="C" OR B4="A" GOTO 73
60 IF (B3="2" AND (B4="1" OR B4="2")) GOTO 73
62 IF B3="3" AND B4="6" GOTO 73
63 IF B3="E" OR B4="9" GOTO 71
```

```
64 IF B3="2" AND B4="3" GOTO 71
65 IF B3="2" AND B4="B" GOTO 72
66 IF B3="A" OR B3="3" GOTO 71
68 IF B4="B" GOTO 73
69 IF B4="3" THEN 73 ELSE 71
70 RUN "SSEXEC3/BAS.DFC:0"
71 RUN "SSEXEC4/BAS.DFC:0"
72 RUN "SSEXEC5/BAS.DFC:0"
73 RUN "SSEXEC6/BAS.DFC:0"
```

APPENDIX J

SINGLE STEP EXECUTION PROGRAM 3

```
1 'SSEXEC3/BAS.DFC:0
2 DEFSTR A-B:DEFINT I-Q
3 M4=PEEK(29652):GOSUB 30:POKE 29652,M4
4 N3=PEEK(29648) AND 1
5 IF M5=2 N3=N3 OR 8
6 IF M6=2 N3=N3 OR 16
7 IF M6=3 N3=N3 OR 32
8 POKE 29648,N3
9 RUN "SSEXEC7/BAS.DFC:0"
30 N1=256*PEEK(29651)+PEEK(29650): M=PEEK(N1): N=M/16: GOSUB
10: B1=B: N=M-N*16: GOSUB 10: B2=B: N=PEEK(29640)
31 IF B1="E" AND B2="9" GOTO 45
32 IF B1="C" AND B2="9" GOTO 46
33 IF B2<>"7" OR B1<>"F" GOTO 36
34 IF B1="C" OR B1="D" OR B1="E" OR B1="F" GOTO 49
36 IF B1="C" AND B2="0" GOTO 51
37 IF B1="D" AND B2="0" GOTO 52
38 IF B1="E" AND B2="0" GOTO 53
39 IF B1="F" AND B2="0" GOTO 54
```

```
40 IF B1="C" AND B2="8" GOTO 55

41 IF B1="D" AND B2="8" GOTO 56

42 IF B1="E" AND B2="8" GOTO 57

43 IF B1="F" AND B2="8" GOTO 58

44 M5=1:M6=3:RETURN

45     M5=1:     M6=1:     POKE    29672,PEEK(29639):     POKE
29673,PEEK(29638): RETURN

46 M4=M4+1:IF M4>127 THEN 47 ELSE 48

47 M5=2:M6=1:RETURN

48    M5=1:     M6=1:    M=256*PEEK(29647)+PEEK(29646):    POKE
29672,PEEK(M+1): POKE  29673,PEEK(M):  POKE  29674,51:  POKE
29675,51: RETURN

49 PRINT "YOU HAVE USED A RST ??? EMULATED WITH A NOP"

50 M5=1:M6=2:RETURN

51 IF (N AND 64 )=64 THEN 50 ELSE 46

52 IF (N AND 1 )=1 THEN 50 ELSE 46

53 IF (N AND 4 )=4 THEN 50 ELSE 46

54 IF (N AND 128)=128 THEN 50 ELSE 46

55 IF (N AND 64 )=0 THEN 50 ELSE 46

56 IF (N AND 1 )=0 THEN 50 ELSE 46

57 IF (N AND 4 )=0 THEN 50 ELSE 46

58 IF (N AND 128)=0 THEN 50 ELSE 46
```

APPENDIX K

SINGLE STEP EXECUTION PROGRAM 4

```
1 'SSEXEC4/BAS.DFC:0
2 DEFSTR A-B:DEFINT I-Q
3 M4=PEEK(29652):GOSUB 30:POKE 29652,M4
4 N3=(PEEK(29648) AND 1) OR 2
5 IF M5=2:N3=N3 OR 8
6 IF M6=2:N3=N3 OR 16
7 IF M6=3:N3=N3 OR 32
8 POKE 29648,N3
9 RUN "SSEXEC7/BAS.DFC:0"
30 N1=256*PEEK(29651)+PEEK(29650): M=PEEK(N1): N=M/16: GOSUB
10: B1=B: N=M-N*16: GOSUB 10: B2=B
31 M=PEEK(N1+1):N=M/16:GOSUB 10:B3=B:N=M-N*16: GOSUB 10:B4=B
32 A1="YOU ARE USING ":A2="  THIS  WILL  BE  EXECUTED  UNTIL
":=PEEK(29640)
33 IF B1="1" AND B2="8" GOTO 53
34 IF B1="3" AND B2="8" GOTO 56
35 IF B1="3" AND B2="0" GOTO 57
36 IF B1="2" AND B2="8" GOTO 58
37 IF B1="2" aND B2="0" GOTO 59
```

```
38 IF B1="D" AND B2="D" AND B3="E" AND B4="9" GOTO 60

39 IF B1="F" AND B2="D" AND B3="E AND B4="9" GOTO 61

40 IF B1="1" AND B2="0" GOTO 62

41 IF B1="E" AND B2="D" THEN 42 ELSE 75

42 IF B3="4" AND B2="D" GOTO 65

43 IF B3="4" AND B2="5" GOTO 66

44 IF B3<>"B" GOTO 75

45 IF B4="0" GOTO 67

46 IF B4="8" GOTO 68

47 IF B4="1" GOTO 69

48 IF B4="9" GOTO 70

49 IF B4="2" GOTO 71

50 IF B4="A" GOTO 72

51 IF B4="3" GOTO 73

52 IF B4="B" THEN 74 ELSE 75

53 IF PEEK(N1+1)>127 GOTO 55

54 N1=N1+PEEK(N1+1)+2:N=N1/256:POKE 29672,N: N=N1-N*256:POKE
29673,N:M5=1: M6=1: RETURN

55        N=256-PEEK(N1+1):N1=N1-N+2:M5=1:M6=1:N=N1/256:POKE
29672,N: N=N1-N*256:POKE 29673,N: RETURN

56 IF (I AND 1 )=0 THEN 76 ELSE 53

57 IF (I AND 1 )=1 THEN 75 ELSE 53

58 IF (I AND 64)=0 THEN 76 ELSE 53

59 IF (I AND 64)=64 THEN 75 ELSE 53

60     M5=1:     M6=1:     POKE     29672,PEEK(29543):     POKE
```

```
29673,PEEK(29642): RETURN

60     M5=1:     M6=1:     POKE     29672,PEEK(29645):     POKE
29673,PEEK(296424: RETURN

62 IF (PEEK(29635)-1)=0 THEN 63 ELSE 64

63 POKE 29674,5:M5=1:M6=2:RETURN

64 POKE 29674,5:IF (PEEK(N1+1) AND 128)=128 THEN 55 ELSE 54

65 PRINT A1,"A RETI ??? EMULATED WITH NOP": GOTO 76

66 PRINT A1," A RETN ??? EMULATED WITH NOP": GOTO 76

67 PRINT A1," LDIR ",A2," BC=0":GOTO 75

68 PRINT A1," LDDR ",A2," BC=0":GOTO 75

69 PRINT A1," CPIR ",A2," BC=0":GOTO 75

70 PRINT A1," CPDR ",A2," BC=0":GOTO 75

71 PRINT A1," INIR ",A2," B=0":GOTO 75

72 PRINT A1," INDR ",A2," B=0":GOTO 75

73 PRINT A1," OTIR ",A2," B=0":GOTO 75

74 PRINT A1," OTDR ",A2," B=0":GOTO 75

75 M5=1:M6=3:RETURN

76 M5=1:M6=2:RETURN
```

APPENDIX L

SINGLE STEP EXECUTION PROGRAM 5

```
1 'SSEXEC5/BAS.DFC:0
2 DEFSTR A-B:DEFINT I-Q
3 M4=PEEK(29652):  GOSUB 30:POKE 29652,M4
4 N3=(PEEK(29648) AND 1) OR 4
5 IF M5=2 N3=N3 OR 8
6 IF M6=2 N3=N3 OR 16
7 IF M6=3 N3=N3 OR 32
8 POKE 29648,N3
9 RUN "SSEXEC7/BAS.DFC:0"
30 N1=256*PEEK(29651)+PEEK(29650): M=PEEK(N1): N=M/16: GOSUB
10: B1=B: N=M-N*16: GOSUB 10: B2=B
31 I=PEEK(29640)
32 IF B2="3" GOTO 50
33 IF B1="C" AND B2="D" GOTO 51
34 IF B1="C" AND B2="2" GOTO 52
35 IF B1="D" AND B2="2" GOTO 53
36 IF B1="E" AND B2="2" GOTO 54
37 IF B1="F" AND B2="2" GOTO 55
38 IF B1="C" AND B2="A" GOTO 56
```

```
39 IF B1="D" AND B2="A" GOTO 57

40 IF B1="E" AND B2="A" GOTO 58

41 IF B1="F" AND B2="A" GOTO 59

42 IF B1="C" AND B2="4" GOTO 60

43 IF B1="D" AND B2="4" GOTO 61

44 IF B1="E" AND B2="4" GOTO 62

45 IF B1="F" AND B2="4" GOTO 63

46 IF B1="C" AND B2="C" GOTO 64

47 IF B1="D" AND B2="C" GOTO 65

48 IF B1="E" AND B2="C" GOTO 66

49 IF B1="F" AND B2="C" THEN 67 ELSE 70

50 M5=1: M6=1: POKE 29672,PEEK(N1+2): POKE 29673,PEEK(N1+1):
RETURN

51      M4=M4-1:      N=N1+3:      M=N/256:      N=N-M*256:
M1=256*PEEK(29647)+PEEK(29646):POKE  M1-1,M:  POKE  M1-2,N:
POKE 29674,59: POKE 29675,59: GOTO 50

52 IF (I AND 64 )=64 THEN 71 ELSE 50

53 IF (I AND 1 )=1 THEN 71 ELSE 50

54 IF (I AND 4 )=4 THEN 71 ELSE 50

55 IF (I AND 128)=128 THEN 71 ELSE 50

56 IF (I AND 64 )=0 THEN 71 ELSE 50

57 IF (I AND 1 )=0 THEN 71 ELSE 50

58 IF (I AND 4 )=0 THEN 71 ELSE 50

59 IF (I AND 128)=0 THEN 71 ELSE 50

60 IF (I AND 64 )=64 THEN 71 ELSE 51
```

```
61 IF (I AND 1 )=1 THEN 71 ELSE 51
62 IF (I AND 4 )=4 THEN 71 ELSE 51
63 IF (I AND 128)=128 THEN 71 ELSE 51
64 IF (I AND 64 )=0 THEN 71 ELSE 51
65 IF (I AND 1 )=0 THEN 71 ELSE 51
66 IF (I AND 4 )=0 THEN 71 ELSE 51
67 IF (I AND 128)=0 THEN 71 ELSE 51
70 M5=1:M6=3:RETURN
71 M5=1:M6=2:RETURN
```

APPENDIX M

SINGLE STEP EXECUTION PORGRAM 6

```
1 'SSEXEC6/BAS.DFC:0
2 DEFINT N
3 N3=((PEEK (29648) AND 1) OR 6) OR 32
4 POKE 29648,N3
5 RUN "SSEXEC7/BAS.DFC:0"
```

APPENDIX N

SINGLE STEP EXECUTION PROGRAM 7

```
1 'SSEXEC7/BAS.DFC:0
2 CLEAR 300:DEFINT I-Q:DEFSTR A-B:DIM A(16)
4 K=PEEK(29649)
5 LP=PEEK(29648) AND 1
6 N3=256*PEEK(29651)+PEEK(29650)
8 IF PEEK(29672)<>0 THEN N1=256*PEEK(29672)+PEEK(29673 :
ELSE N1=N3
10 I=PEEK(29648)
14 if (i and 8)=8 then m5=2 else m5=1
16 IF (I AND 48)=0 THEN M6=1
18 IF (I AND 48)=16 THEN M6=2
20 IF (I AND 48)=32 THEN M6=3
22 IF (I AND 6)=0 THEN N2=1
24 IF (I AND 6)=2 THEN N2=2
26 IF (I AND 6)=4 THEN N2=3
28 IF (I AND 6)=6 THEN N2=4
30 POKE 29672,0:POKE 29673,0
32 ON M6 GOTO 41,40,34
34 FOR K1=29672 TO 29671+N2
```

```
36 POKE K1,PEEK(N1):N1=N1+1
38 NEXT K1
40 N1=N3+N2
41 N=N1/256: POKE 29651,N: N=N1-N*256: POKE 29650,N
42 DEFUSR0=29653:X=USR0(0)
44 N=N3:O=4:GOSUB 100:A(16)=B
46 FOR K1=0 TO 13
48 N=PEEK(29634+K1):O=2:GOSUB 100:A(K1+2)=B
50 NEXT K1
52 FOR K1=0 TO 13
54 A(K1)=A(K1+2)
56 NEXT K1
58 A(13)=A(13)+A(12):A(9)=A(9)+A(8):A(11)=A(11)+A(10)
60 B=A(16)+" "+A(6)+" "+A(7)+"  "+A(1)+"  "+A(0)+"  "+A(3)+"
"+A(2)+" "+A(5)+" "+A(4)+" "+A(13)+" "+A(9)+" "+A(11)
62 IF LP=0 PRINT B
64 IF LP=1 LPRINT B
66 IF M5v2 GOTO 70
68  K=K+1: POKE 29649,K: IF K>9 THEN RUN "SSEXEC1/BAS.DFC:0"
ELSE RUN "SSEXEC2/BAS.DFC:0"
70 PRINT "END OF EXECUTION"
72 RUN "MONITOR/BAS.DFC:0"
```

APPENDIX O

SINGLE STEP EXECUTION MACHINE LANGUAGE PROGRAM

LOC DEC	LOC HEX	HEX CODE	MNEMONIC	Comments
29653	73D5	ED73C073	LD(nn),SP	Save the basic language sp
29657	73D9	31C273	LD SP,nn	Set stack pointer
29660	73DC	C1	POP BC	
29661	73DD	D1	POP DE	
29662	73DE	E1	POP HL	
29663	73DF	F1	POP AF	
29664	73E0	DDE1	POP IX	
29666	73E2	FDE1	POP IY	
29668	73E4	ED7BCE73	LD SP(nn)	
	73E8			
	73E9			
	73EA			
	73EB			
29676	73EC	ED73CE73	LD(nn),SP	

LOC DEC	LOC HEX	HEX CODE	MNEMONIC	Comments
29680	73F0	31CE73	LD SP,nn	
29683	73F3	FDE5	PUSH IY	
29685	73F5	DDE5	PUSH IX	
29687	73F7	F5	PUSH AF	
29688	73F8	E5	PUSH HL	
29689	73F9	D5	PUSH DE	
29690	73FA	C5	PUSH BC	
29691	73FB	ED7BC073	LD SP(nn)	
29695	73FF	C9	RET	

APPENDIX P

DSTORE PROGRAM

```
700 'DSTORE: A PART OF A Z-80 MICROPROCESSOR DEVELOPMENT AID
FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

701 DEFSTR A-B:DEFINT I-O

705 PRINT "DSTORE MODE - TO STORE A PROGRAM ON THE DISK"

710 INPUT "STARTING MEMORY LOCATION IN DECIMAL";N1

715 INPUT "ENDING MEMORY LOCATION IN DECIMAL";N2

720 INPUT "INPUT FILENAME (NO MORE THAN 8 ALPHA-NUMERIC
CHARACTERS AND ANY OPTIONAL PARAMETERS)";A

725 OPEN "O",1,A

730 FOR N=N1 TO N2

735 PRINT#1,PEEK(N)

740 NEXT N

745 CLOSE 1

750 PRINT "PROGRAM DATA IS STORED IN ",A," NOTE THAT"

755 RUN "MONITOR/BAS.DFC:0"
```

APPENDIX Q

DRECALL PROGRAM

```
800  'DRECALL:  A  PART OF A Z-80 MICROPROCESSOR DEVELOPMENT
AID FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

805 DEFSTR A-B:DEFINT N

810 PRINT "DRECALL MODE - TO RECALL A PROGRAM FROM THE DISK"

815 INPUT "STARTING MEMORY LOCATION IN DECIMAL";N1

820 INPUT "ENDING MEMORY LOCATION IN DECIMAL";N2

825 INPUT "FILENAME AS USED DURING DSTORE";A

830 OPEN "I",1,A

835 FOR N=N1 TO N2

840 INPUT#1,N3

845 POKE N,N3

850 NEXT N

855 CLOSE 1

860 PRINT "THE PROGRAM HAS BEEN LOADED INTO MEMORY LOCATIONS
",N1," TO ",N2

865 RUN "MONITOR/BAS.DFC:0"
```

APPENDIX R

TSTORE PROGRAM

```
700'TSTORE:  A PART OF A Z-80 MICROPROCESSOR DEVELOPMENT AID
FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

710 DEFSTR A-B:DEFINT I-N:DIM I(15)

720 PRINT "TSTORE MODE TO STORE A PROGRAM ON  TAPE  RECORDER
#1"

730 INPUT "STARTING MEMORY LOCATION IN DECIMAL";N1

740 INPUT "ENDING MEMORY LOCATION IN DECIMAL";N2

745 PRINT "NOTE  STARTING  TAPE  COUNTER  AND  TURN  ON  THE
RECORDER TO RECORD"

750 FOR J=0 TO 15

755 I(J)=PEEK(N1):N1=N1+1

760 NEXT J

765   PRINT#-1,I(0),I(1),I(2),I(3),I(4),I(5),I(6),I(7),I(8),
I(9),I(10),I(11),I(12),I(13),I(14),I(15)

770 IF N1<N2 GOTO 750

775 PRINT "NOTE ENDING TAPE COUNTER"

780 INPUT "CONTINUE TSTORE MODE (YES OR NO)";B

785 IF B="YES" THEN 720 ELSE RUN "MONITOR/BAS.DFC:0"
```

APPENDIX S

TRECALL PROGRAM

```
800  'TRECALL:  A  PART OF A Z-80 MICROPROCESSOR DEVELOPMENT
AID FOR DISK BASIC ON THE TRS-80 BY DF CORTEVILLE APR 79

805 DEFSTR A-B:DEFINT I-N:DIM I(15)

810 PRINT "TRECALL MODE TO RECALL A PROGRAM  FROM  THE  TAPE
RECORDER"

815 INPUT "STARTING MEMORY LOCATION IN DECINAL";N1

820 INPUT "ENDING MEMORY LOCATION IN DECIMAL";N2

825 PRINT "TURN ON THE TAPE RECORDER (PLAY)  AT  THESTARTING
COUNTER"

830 INPUT "WHEN TAPE IS READY PRESS ENTER";A

850   INPUT#-1,I(0),I(1),I(2),I(3),I(4),I(5),I(6),I(7),I(8),
I(9),I(10),I(11),I(12),I(13),I(14),I(15)

855 FOR J=0 TO 15

860 IF N1>N2 GOTO 870

865 POKE N1,I(J):N1=N1+1

870 NEXT J

875 IF N1>N2 THEN 850 ELSE RUN "MONITOR/BAS.DFC:0"
```

APPENDIX T

SUBROUTINE 10 DECIMAL TO HEX CONVERSION

```
10 IF N=0 B="0"
11 IF N=1 B="1"
12 IF N=2 B="2"
13 IF N=3 B="3"
14 IF N=4 B="4"
15 IF N=5 B="5"
16 IF N=6 B="6"
17 IF N=7 B="7"
18 IF N=8 B="8"
19 IF N=9 B="9"
20 IF N=10 B="A"
21 IF N=11 B="B"
22 IF N=12 B="C"
23 IF N=13 B="D"
24 IF N=14 B="E"
25 IF N=15 B="F"
26 RETURN
```

APPENDIX U

SUBROUTINE 100 DECIMAL TO HEX CONVERSION

```
110 M=16:M1=256:M2=4096:IF O=4 GOTO 115
105 IF O=2 GOTO 120
115     I(0)=N/M2:N=N-I(0)*M2:I(1)=N/M1:N=N-I(1)*M1:I(2)=N/M:
I(3)=N-I(2)*M:GOTO 130
120 I(0)=N/M:I(1)=N-I(0)*M
130 FOR J=0 TO O-1
140 IF I(J)=0 A(J)="0"
141 IF I(J)=1 A(J)="1"
142 IF I(J)=2 A(J)="2"
143 IF I(J)=3 A(J)="3"
144 IF I(J)=4 A(J)="4"
145 IF I(J)=5 A(J)="5"
146 IF I(J)=6 A(J)="6"
147 IF I(J)=7 A(J)="7"
148 IF I(J)=8 A(J)="8"
149 IF I(J)=9 A(J)="9"
150 IF I(J)=10 A(J)="A"
151 IF I(J)=11 A(J)="B"
152 IF I(J)=12 A(J)="C"
```

```
153 IF I(J)=13 A(J)="D"
154 IF I(J)=14 A(J)="E"
155 IF I(J)=15 A(J)="F"
156 NEXT
170 IF O=2 THEN B=A(0)+A(1)
180 IF O=4 THEN B=A(0)+A(1)+A(2)+A(3)
190 RETURN
```

APPENDIX V

SUBROUTINE 200 HEX TO DECIMAL CONVERSION

```
220 A(0)=LEFT$(B,1):A(1)=MID$(B,2,1):M3=16
210 FOR J=0 TO 1
220 IF A(J)="0" I(J)=0
221 IF A(J)="1" I(J)=1
222 IF A(J)="2" I(J)=2
223 IF A(J)="3" I(J)=3
224 IF A(J)="4" I(J)=4
225 IF A(J)="5" I(J)=5
226 IF A(J)="6" I(J)=6
227 IF A(J)="7" I(J)=7
228 IF A(J)="8" I(J)=8
228 IF A(J)="8" I(J)=8
229 IF A(J)="9" I(J)=9
230 IF A(J)="A" I(J)=10
231 IF A(J)="B" I(J)=11
232 IF A(J)="C" I(J)=12
233 IF A(J)="D" I(J)=13
234 IF A(J)="E" I(J)=14
```

```
235 IF A(J)="F" I(J)=15
236 IF A(J)=" " I(J)=0
240 NEXT
250 N=M3*I(0)+I(1)
260 RETURN
```

APPENDIX W

SINGLE STEP EXECUTION MEMORY MAP

DEC	HEX	Contents
29632	73C0	Basic language stack pointer
29634	73C2	C register
29635	73C3	B register
29636	73C4	E register
29637	73C5	D register
29638	73C6	L register
29639	73C7	H register
29640	73C8	F register
29641	73C9	A register
29642	73CA	IX register
29644	73CC	IY register
29646	73CE	Stack pointer
29648	73D0	D0 is LP
		D1 and D2 are N2
		D3 is M5
		D4 and D5 are M6
29649	73D1	K

DEC	HEX	Contents
29650	73D2	N1 (PClow)
29651	73D3	N1 (PChigh)
29652	73D4	M4
29653	73D5	Machine language program
to	to	
29671	73E7	Appendix D
29672	73E8	Four bytes for
to	to	
29675	73EB	Executed step
29676	73EC	Machine language program
to	to	
29695	73FF	Appendix D

APPENDIX X

LAB 1

1. Place a chair in front of the TRS-80.
2. Check six 60hz AC power cords (two from the expansion interface, one from video display, one from the tape recorder, one from the mini disk drive, and one from the line printer interface).
3. Check that the video display is connected to the micro computer system (keyboard) in the plug marked video (center plug).
4. Check that the tape recorder plug is connected to the keyboard at the plug marked tape; and to the tape recorder as follows: black to EAR, large gray to AUX, and small gray to REMOTE.
5. Check that the keyboard is connected to the expansion interface with a forty pin edge connector and a power cord to the plug marked power on the keyboard.
6. Check that the line printer interface is connected to the expansion interface with a forty pin edge connector.
7. Check that the mini disk drive is connected to the expansion interface.
8. Place the line printer interface on/off switch to on then press the reset push button. (note: If the line printer interface is off, then the mini disk drive and expansion interface will not work.)

9. Turn on the video display. Press the power switch in.

10. Place the operating system mini disk, a Z-80 microprocessor development aid, in the mini disk drive. The catalog number should be showing on the bottom right. Close the mini disk drive door.

11. Turn on the mini disk drive. On is up on the switch on the back of the mini disk drive.

12. Turn on the expansion interface. Push in the button switch on the center front of the expansion interface.

13. Turn on the keyboard. Push in the button switch on the rear right of the keyboard. The mini disk drive should turn on thus loading the TRSDOS (TRS disk operating system) into the RAM. The following appears on the video display:

    TRSDOS - DISK OPERTING SYSTEM - VER 2.1

    DOS READY

14. Type BASIC and push ENTER.

15. For the question, HOW MANY FILES?, press ENTER. This defaults on three files.

16. For the question, MEMORY SIZE?, type 29630 and press ENTER.

17. RADIO SHACK DISK BASIC VERSION 1.1

    READY

    Appears on the video display, type RUN "S" and press ENTER.

18. MONITOR PROGRAM

    A Z-80 MICROPROCESSOR DEVELOPMENT AID

SHORT OR LONG EXPLANATION? INPUT S OR L?

Appears on the video display, type L and press ENTER.

19. The long explanation for each mode appears on the video display, type S and press ENTER.

20. The short explanation appears on the video display, type DRECALL and press ENTER.

21. The disk recall program is now asking for a beginning memory location to load a program into RAM from the disk, type 29696 and press ENTER.

22. The ending memory location is required, type 29710 and press ENTER.

23. A filename is required, type TESTPR#1 and press ENTER.

24. Steps 22 to 24 are unique for this lab and are dependent on the program to be loaded. The monitor program is requesting a long or short explanation. Type L and press ENTER.

25. Examine the program just entered into the RAM, type DUMP and press ENTER.

26. For starting memory location, type 29696 and press ENTER.

27. For the number of memory locations, type 15 and press ENTER.

28. For video output type V and press ENTER. This shows the beginning decimal memory location, the beginning hexadecimal memory location and the contents of the following 16 memory locations.

29696 7400 21 00 3C 3E BF 06 02 77 23 05 C8 C3 07 74 FF FF

29. Type NO and press ENTER. This returns to the monitor program.

30. Type S and press ENTER.

31. The LOAD mode will be used to change location 7406 from 02 to FF. Type LOAD and press ENTER.

32. Type 29702 and press ENTER.

33. Type FF and press ENTER.

34. Exit the LOAD mode, type QUIT and press ENTER. This returns to the monitor program.

35. Type S and press ENTER.

36. Repeat steps 26 to 29. The 02 in memory location 7406 is now FF.

37. Press ENTER (no is the default). This returns to the monitor program.

38. Type S and press ENTER.

39. Type EXEC and press ENTER to execute the test program.

40. Type 29696 and press ENTER for the starting memory location.

41. The top four lines of the video display are whitened out, 256 video memory locations.

42. Press ENTER (no is the default). This returns to the monitor program.

43. Type S and press ENTER.

44. Type END and press ENTER. Secure all power supplies. This ends lab 1.

TESTPR#1

MEM LOC	OPCODE	MNEMONIC
7400	21003C	LD HL, 3C00

```
7403      3EBF      LD A,BF
7405      06FF      LD B,FF
7407      77        LD (HL),A
7408      23        INC HL
7409      05        DEC B
740A      C8        RET Z
740B      C30774    JP 7407
```

APPENDIX Y

LAB 2

1. Power up the system by completing steps 1 through 18 of lab 1.
2. Type S and press ENTER.
3. Recall test program number 4 from the disk: type DRECALL and press ENTER.
4. Type 29696 and press ENTER for the starting memory location.
5. Type 29770 and press ENTER for the ending memory location.
6. Type TESTPRO4 and press ENTER for the filename.
7. Type S and press ENTER.
8. Type DUMP and press ENTER.
9. Type 29696 and press ENTER for the starting memory location.
10. Type 74 and press ENTER for the number of memory locations.
11. Connect the cable from the line printer interface to the line printer and turn on the printer.
12. Type LP and press ENTER to make a hardcopy of the program. There is a 20 second delay between pressing ENTER and the first output which will look like this:

```
29696 7400 AF 31 00 75 3E BF 21 F8 3F 77 CD 10 74 77 C9 00
29712 7410 06 02 05 00 CA 00 74 05 77 CC 20 74 77 C9 00 00
29728 7420 06 02 77 05 20 FD CD 30 74 77 C9 00 00 00 00 00
29744 7430 DD 21 3F 74 FD 21 47 74 77 DD E9 77 00 FD E9 06
29760 7440 03 77 10 F7 C3 3B 74 77 C9 00 00 00 00 00 00 00
```

13. Press ENTER,NO is the default.

14. Type S and press ENTER.

15. Type SS EXEC and press ENTER. This starts the single step execution mode.

16. Type V and press ENTER.

17. Type 29696 and press ENTER for starting memory location.

18. Single step execution requires about 20 seconds for each step. Use this time to anticipate the next step. The printout produced in step 12 or the program at the end of this lab should be used.

19. Note the four levels of calls and the use of the video RAM location 3FF8 as a LED.

20. After end of execution, type S and press ENTER.

21. Type END and press ENTER.

22. This ends lab 2. If the student needs more practice, then redo lab 2 using LP instead of V in step 16 to obtain a hard copy of the execution.

TEST PROGRAM #4

MEM LOC	OPCODE	MNEMONIC
7400	AF	XRA A

```
7401      310075    LD SP, 7500
7404      3EBF      LD A,BF
7406      21F83F    LD HL, 3FF8
7409      LD (HL), A
740A      CD1074    CALL 7410
740D      77        LD (HL),A
740E      C9        RET
740F      00        NOP
MEM LOC   OPCODE    MNEMONIC
7410      0602      LD B, 02
7412      05        DEC B
7413      00        NOP
7414      CA0074    JPZ 7400
7417      05        DEC B
7418      77        LD (HL),A
7419      CC2074    CZ 7420
741C      77        LD (HL),A
741D      C9        RET
741E      00        NOP
741F      00        NOP
MEM LOC   OPCODE    MNEMONIC
7420      0602      LD B, 02
7422      77        LD (HL), A
7423      05        DEC B
```

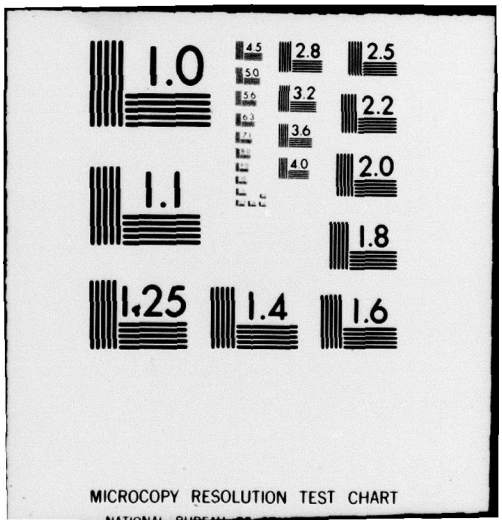
AD-A073 927 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
THE DESIGN AND IMPLEMENTATION OF AN INEXPENSIVE MICROPROCESSOR --ETC(U)
JUN 79 D F CORTEVILLE

UNCLASSIFIED NL

2 OF 2
ADA
073927



END
DATE
FILMED
10 -79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
MICROCOPY RESOLUTION TEST CHART

```
7424        20FD        JRNZ FD
7426        CD3074      CALL 7430
7429        77          LD (HL), A
742A        C9          RET
742B        00          NOP
742C        00          NOP
742D        00          NOP
742E        00          NOP
742F        00          NOP
MEM LOC     OPCODE      MNEMONIC
7430        DD213F74    LD IX, 743F
7434        FD214774    LD IY 7447
7438        77          LD (HL), A
743A        DDE9        JP(IX)
743B        77          LD (HL),A
743C        00          NOP
743D        FDE9        JP(IY)
743F        0603        LD B,03
MEM LOC     OPCODE      MNEMONIC
7441        77          LD (HL), A
7442        10F7        DJNZ F7
7444        C33B74      JP 743B
7447        77          LD (HL), A
7448        C9          RET
```

APPENDIX Z

LAB 3

1. Power up the system by completing steps 1 through 18 of lab 1.
2. Type S and press ENTER.
3. Type LOAD and press ENTER to load your machine language program into the RAM.
4. Type the starting memory location in decimal and press ENTER. Protected memory locations are 29695 to 33777. Use the machine language program at the end of this lab if you did not write one prior to this lab session.
5. Type the opcode one memory location at a time and press ENTER. For each memory location the system displays the memory address and the previous contents. If ENTER is pressed without data, then the memory content is not changed.
6. Type QUIT and press ENTER when the complete program is entered.
7. Type S and press ENTER.
8. Type DUMP and press ENTER.
9. Type the starting memory location in decimal and press ENTER.
10. Type the number of memory locations in decimal and press ENTER.

11. Type V or LP for video or lineprinter and press ENTER.

12. Compare the output with the desired program.

13. Debug the program using the execution mode as used in lab 1 and the single step execution mode as used in lab 2.

14. Use the load and dump modes to change and verify the program if necessary.

15. Save the machine language program on the disk drive. From the monitor program type DSTORE and press ENTER.

16. Type the starting memory location in decimal and press ENTER.

17. Type the ending memory location in decimal and press ENTER.

18. Type a filename (your initals will be fine and easy to remember) and press ENTER.

19. Type S and press ENTER.

20. Save the machine language program on the tape recorder. Place a tape in the tape recorder, remember the tape counter number, and press the record and play buttons simultaneously.

21. Type TSTORE and press ENTER.

22. Type the starting memory location in decimal and press ENTER.

23. Type the ending memory location in decimal and press ENTER.

24. Note the ending tape counter number. Press ENTER, NO is the default.

25. Press the stop button on the tape recorder.

26. Type S and press ENTER.

27. Retrieve the machine language program from the tape recorder. Rewind the tape recorder to the starting tape counter number. Place the tape recorder in play.

28. Type TRECALL and press ENTER.

29. Pick a new starting memory location. Type the new starting memory location and press ENTER.

30. Type the new ending memory location and press ENTER.

31. The tape recorder should be ready so prsss ENTER.

32. Type S and press ENTER.

33. Use the dump mode to verify the machine language program retrieved from the tape recorder.

34. Secure the system from the monitor program. Type END and press ENTER.

35. This ends lab 3. The student has operated every mode of the Z-80 microprocessor development aid. For a discussion of the system read The Design and Implementation of an Inexpensive Microprocessor Development System for the Z-80 Microprocessor, a thesis, by D. F. Corteville June 1979.

SAMPLE PROGRAM

MEM LOC	OPCODE	MNEMONIC
7400	1802	JR 02
7402	00	NOP
7403	00	NOP
7404	3EAA	LD A, AA

```
7406        1805        JR 05

7408        00          NOP

7409        00          NOP

740A        06BB        LD B, BB

740C        00          NOP

740D        00          NOP

740E        00          NOP

740F        C9          RET
```

LIST OF REFERENCES

1. Radio Shack, TRS-80 Microcomputer Technical Reference Handbook, 1978.

2. Jim-pak, Microprocessor/LED Data Book.

3. Jim-pak, 7400/74LS Data Book.

4. Jim-pak, CMOS/Linear Data Book.

5. Radio Shack, Level II Basic Reference Manual, TRS-80 Micro Computer System, 1978.

6. Radio Shack, Preliminary Instruction Manual Disk Basic Version 1.1 TRSDOS Version 2.1, 1978.

7. David L. Cohn and James L. Melas, A Step by Step Introduction to 8080 Microprocessor Systems, Dilithium Press, 1977.

8. William Barden, Jr, The Z-80 Microcomputer handbook, Sams Publication, 1978.

9. Radio Shack, T-BUG Z-80 Monitor and Debugging Aid, 1978.

10. Radio Shack, TRS-80 Editor/Assembler, 1978.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defence Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Professor R. Panholzer, Code 62Pz Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
5.	Associate Professor M. L. Cotton, Code 62Co Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	LT Douglas F. Corteville, USN 2215 Fairplain Avenue Benton Harbor, Michigan 49022	1