1-1-1981

The implementation of a multi-user facility for microprocessor software development.

Charles Hagel Kercsmar

THE IMPLEMENTATION OF A MULTI-USER FACILITY FOR MICROPROCESSOR SOFTWARE DEVELOPMENT

by

Charles Hagel Kercsmar

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

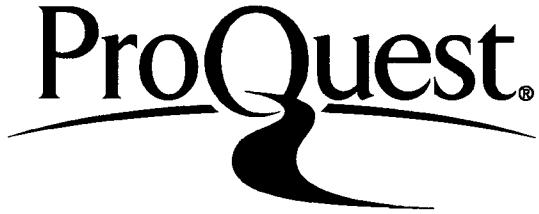
Master of Science

in

Computing Science

Lehigh University

1981



This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 5, 1981
date

Professor in Charge

Head of Division

ACKNOWLEDGEMENTS

The author wishes to acknowledge the beneficial advice provided by Mr. Gary D. Crouse, Bethlehem Steel Corporation Research Department, during numerous discussions concerning microprocessor implementation methodology.

A further acknowledgement is made to the Bethlehem Steel Corporation Research Department for permitting use of microprocessor development facilities where a substantial portion of the work described in this thesis was performed.

TABLE OF CONTENTS

A B S T R A C T

The economic advantages associated with large scale integration of semi-conductor electronic devices have introduced opportunities for wide-spread application of microprocessor computer systems. It is essential that Computer Science students be familiar with microprocessor organization and capabilities, and be able to design and implement software systems for microprocessor application. This thesis describes efforts made by the author to provide microprocessor tools which might enhance the learning experience of a student by allowing pragmatic, hands-on exercises in a user-friendly environment. Two major achievements were accomplished during this work. The first area of effort was to install a cross-assembler for a microprocessor source language. This cross-assembler was installed on the Lehigh DECSYSTEM-20 computer. Since most students would probably have prior exposure to the DEC-20 before undertaking a microprocessor course, the learning curve for using the cross-assembler is expected to be minimized. The second major achievement was to design and implement a communications interface between a microprocessor computer and the DECSYSTEM-20 for the purpose of providing an interactive capability from a

terminal attached to the microprocessor. The interface is designed to work in two modes. In a purely interactive mode, all key entries are passed through the microprocessor to the DEC-20 and all communication from the DEC-20 is output to the terminal. When the microprocessor is not in the interactive mode but the communications link is established, the user may request microprocessor object files to be transmitted from the DEC-20 to the microprocessor at which point the microprocessor software loads the user programs into random access storage (RAM). The user programs may then be requested to run via terminal input. A number of service routines are also available within the microprocessor software to facilitate ease of use.

INTRODUCTION

This thesis describes an integrated complement of computer hardware and software which was configured to provide a microprocessor development system for student use. A major objective was to use existing facilities and software where possible and to supplement and modify these facilities as necessary for the enhancement of a user-friendly environment. The work proceeded in two major phases. During the first phase, it was necessary to select a microprocessor which would be appropriate for such a project. The Intel 8080-8085 microprocessor was chosen as a result of two major considerations. First, the Intel 8080 has received wide acceptance within the user community. On this basis, knowledge of the 8080 would be more universally applicable than any other potential candidate. Secondly, in order to provide a development facility, a cross-assembler for the selected microprocessor source language was required. Lehigh had previously acquired a cross-assembler for the Intel 8080-8085 Assembly Language, and had the assembler installed on the Control Data Corporation (CDC) 6400 computer system. Selection of the 8080 then precluded

the need for purchasing any additional cross-assembler software.

Once the Intel 8080 had been identified as the target microprocessor for the development system, the first phase of the project continued with the installation of the cross-assembler on the DECSYSTEM-20 computer and the establishment of procedures for its use.

The second phase of the project was initiated with the identification of an Intel 8080-8085 based single board computer which would provide exercising hardware for user programs. Requisites for the computer included adequate on-board programmable read only memory (PROM), adequate random access memory (RAM) and a minimum of two serial input-output ports. The Intel iSBC 544 Intelligent Communications Controller Board was chosen as the most cost-effective alternative which met the above requirements. Executive software installed on the Intel 544 computer includes a repertoire of user utilities and service routines as well as a communications interface to the DECSYSTEM-20 computer system. A description of the microprocessor development system hardware and software

characteristics and a discussion of system use is presented in the following sections.

HARDWARE DESCRIPTION

The hardware configuration required for the microprocessor development system consists of four key elements, namely the DECSYSTEM-20 computer system including attached peripherals, the Intel iSBC 544 Communications Controller Single Board Computer with power supplies, a terminal device capable of interfacing to the Intel 544 board, and a modem which is compatible with those attached to the DECSYSTEM-20. A description of each element and the functional role it performs within the context of the microprocessor development system follows.

DECSYSTEM-20 FUNCTIONAL AND HARDWARE DESCRIPTION

The DECSYSTEM_20 serves as the host hardware for microprocessor source code preparation and assembly. The multi-user interactive aspects of the DECSYSTEM-20

---

* See references, Page 33 .

(1)* makes it particularly suitable for a large population of users, thus avoiding a common problem of system access frequently encountered with single user microprocessor development systems.

The complement of hardware which comprises the DECSYSTEM-20 includes:

- o DEC 2040 Central processor unit, 256K 36 bit words
- o Line printer, 230 lines per minute
- o Tape drive, 9-track, 800(NRZI) & 1600(PE) densitiy
- o Disk drives (2), 100M characters each
- o Terminal ports (48), 1200 and 110/300 baud

Microprocessor source code preparation is performed using the DECSYSTEM-20 text editor. The interactive features available on the DECSYSTEM-20 afford the conveniences of editing ease and file storage and management to the user. The Intel 8080-8085 cross-assembler is also resident on the DEC-20. The preparation of an error-free microprocessor program would typically involve several iterations of source code editing and assembly to produce an object file which is capable of being loaded into and exercised on the microprocessor hardware.

INTEL ISBC 544 INTELLIGENT COMMUNICATIONS CONTROLLER BOARD FUNCTIONAL AND HARDWARE DESCRIPTION

The iSBC 544 Intelligent Communications Controller (2)* is a member of the Intel line of single board computers. It is capable of operating as an intelligent slave within a distributed processing network but in this project was configured and used as a stand-alone master communicating with the DECSYSTEM-20 via a 300 baud modem over a dial-up phone line. The processor on the iSBC 544 board is an 8085A CPU. Also included on the 544 board is 16K bytes of dynamic random access memory, 8K bytes of programmable read-only memory, 4 serial synchronous/asynchronous input/output ports, and 1 parallel input/output port. Baud rates, data formats, and interrupt priorities are individually selectable for each port on the board. Other features available on the iSBC 544 include programmable interval timers and a programmable interrupt controller.

---

* See references, Page 33 .

The iSBC 544 is designed to interface with the standard Intel backplane, commonly referred to as the multibus. The 544 can operate without the backplane, however, provided power requirements of +5v, +12v, and -12v are available.

TERMINAL EQUIPMENT FUNCTIONAL AND HARDWARE DESCRIPTION

The terminal equipment which attaches to the iSBC 544 computer must conform in spirit to the EIA RS232 standard. Many terminals are available which would be suitable for use, including both CRT and printing varieties. The terminal equipment used during the development of this project was a Texas Instruments Silent 700 series device. The interface to the iSBC 544 was operated at 300 baud, 1 stop bit, parity disabled, 8 bit character transfer. If a particular terminal device could not support these parameters, however, the interface characteristics could easily be modified in software to handle any conflicts. The transfer rate of 300 baud was retained to prevent the thermal printer of the terminal from being overdriven. If a CRT or faster

printing terminal had been implemented as the system console, the baud rate could have been increased.

The terminal provides input/output capability for user data entry and display to both the iSBC 544 and, provided communications are established, to the DEC-20. The description of available user entered commands are presented in subsequent sections of this document.

MODEM DEVICE FUNCTIONAL AND HARDWARE DESCRIPTION

The modem equipment required for the microprocessor development system must be compatible with those attached to the DECSYSTEM-20. During the course of this project, all communications to the DEC-20 was performed over dial-up lines at 300 baud. This communications medium was chosen in order to preclude any delay which might have resulted from awaiting permission to attach the iSBC 544 to the DEC-20 via a dedicated line over a short haul modem. This later configuration would be more desirable should the microprocessor development system be put to heavy use.

The system would be fully compatible with this enhancement and would allow large data transfer to proceed at 1200 baud.

The modem used for the development work was manufactured by Digicom Data Products. It is a 300 baud acoustically coupled data set, with an EIA RS232 interface for connection to the iSBC 544 computer. The modem operates at 115 V AC power. An indicator light on the modem signals when the carrier from the DEC-20 is detected as valid. A second indicator light can be observed to detect data transmission in either direction on the communications line.

SOFTWARE DESCRIPTION

Two major software components are included as components of the microprocessor development system. The first of these is the Intel 8080-8085 Assembly Language cross-assembler (3)* . The cross-assembler is designed to be hosted on any general-purpose digital computer with sufficient memory and an integer size of 30 bits or more. The cross-assembler was developed by Intel and dubbed MAC80. The source language of the assembler is ANSI Standard FORTRAN (1966), and therefore is quite transportable to most machines with a minimum number of required changes, usually only arising from input and output peculiarities. The assembler is designed to transform Intel Assembly Language (4)* source programs into ASCII code hexidecimal machine language object modules.

The second software component is the Microprocessor Development Executive. This program provides the supervisory control logic for the iSBC 544 computer.

---

* See references, Page 33 .

This executive was cloned from another executive, RMUX (5)* , which was designed by Gary D. Crouse (6)* of the Bethlehem Steel Corporation Research Department. Significant modifications were made to the interrupt structure of RMUX to incorporate interrupt driven input and output from the system console and the DEC-20 interface. Also included in the executive is the DEC-20 communications logic and numerous user service routines which can be invoked both by user programs and from the system console. A discussion of these two software components follows.

MAC80 CROSS-ASSEMBLER FUNCTIONAL DESCRIPTION

The MAC80 Macro Assembler (3) * translates a symbolic representation of 8080-8085 microcomputer instructions and data into a form which is an ASCII code representation of the machine instructions executable by the 8080 or 8085 microcomputer. The input to the MAC80 cross-assembler is in the form of macro

---

* See references, Page 33 .

definitions, instruction mneumonics, and symbolic definitions and references. The output is in object code form, including starting memory location, byte count, hexidecimal machine code, and checksum.

MAC80 Cross-assembler General Characteristics

The MAC80 cross-assembler (3)* is a stand-alone program designed to be hosted on most general purpose digital computer systems. It can be run in a batch mode or interactively. A source file containing Intel 8080-8085 Assembly language (4)* statements must be prepared prior to invoking the cross-assembler. During execution, the cross-assembler produces a listing file containing the source statements and the resulting generated hexidecimal machine code. Error messages are annotated on the listing to identify those source statements which have been flagged by the cross-assembler. A symbolic reference list may be

---

* See references, Page 33 .

optionally appended to the listing file. Also generated during execution is an object code file. The object code file is in a form suitable for loading into the 8080-8085 based microcompouter.

The MAC80 cross-assembler is written in ANSI Standard FORTRAN IV, so that it might easily be transported from one host computer to another. In some instances, speed of execution was sacrificed to meet the above feature. Close examination of the source code may identify potential areas for speed improvement if required. A symbol table within MAC80 is dimensioned to 1000 integer words, with each symbol requiring two words. This table can expanded, memory permitting, by a minor programming change within MAC80, should the need arise. Source listings of MAC80 in both machine readable and printed form are available from the Computing and Information Science Department.

MAC80 Input-Output File Descriptions

Five files are referenced during the execution of the MAC80 cross-assembler. These files include the interactive terminal interface, the source assembler code file, the object code output file, the listing output file, and the MAC80 environment control file. A description of the content and format of each of the files is presented in the following sections.

MAC80 Interactive Terminal File Description

The MAC80 Interactive Terminal file presents real-time status of the assembly process to the user. When the execution of MAC80 is envoked by user entry of the run command, the user is assured of execution commencement by receipt of the MAC80 version banner and "BEGIN" message. At the completion of MAC80 execution, a summary error count is presented to the user together with an "EXIT" message. The Interactive Terminal file is defined as logical unit 5 within the MAC80 cross-assembler program.

MAC80 Source Assembly Language Code File Description

The MAC80 source file contains the 8080-8085 Assembly Language source statements which are to be compiled by the cross-assembler. The source file is defined as logical unit 20 within the MAC80 program, and is referenced as file FOR20.DAT within the DECSYSTEM-20 environment. The source file must be created using the DECSYSTEM-20 editor prior to envoking the MAC80 cross-assembler. Each line of the file contains a single source code statement, with each statement being either an assemblable statement or a comment statement. The fields of an assemblable statement are free format with a fixed order of occurrance.

The ordering of fields, which must be adhered to, is as follows:

1. Label (optional)
2. Operation code
3. Operand(s)
4. Comments (optional)

The label field is terminated with a colon. Comments commence with a semi-colon. All fields

are delimited with either one or more blanks, a colon, or a semi-colon. A comment statement is one which a semi-colon appears as the first non-blank character.

MAC80 Listing File Description

The MAC80 Listing file is created during MAC80 execution and is formatted with standard control characters to be output to a printer device. The listing file is written to logical device 21 which corresponds to file FOR21.DAT in the DECSYSTEM-20 environment.

The listing file format is fixed and contains the following information:

- Assembler error code
- Program location counter value
- Nesting level for IF-ENDIF and macros
- Generated object code
- Macro expansion flag
- Source assembly statement

MAC80 Object File Description

The MAC80 Object file is created during execution and is written to logical device 22, or FOR22.DAT within the DEC-20 system. The format of the object file is user selectable via control switch input to MAC80. However, for the microprocessor development system purposes, the hexidecimal object code format is required. In this format, the records are an ASCII representation of program memory with additional fields for start of record indicator, starting address, byte count, and checksum.
The format layout is as follows:

COLUMN(S)	DESCRIPTION
1	Colon start of record indicator
2 - 3	Hexidecimal record length
4 - 7	Hexidecimal memory address
8 - 9	Record type (not implemented)
10 - N-1(*)	Data
N - N+1(*)	Checksum

* N = 10 + (2 x record length)

The record length is the count of actual data bytes in the record. The checksum is the negative of the sum of all 8-bit bytes in the record, evaluated modulo 256. The sum of all bytes in the record, including the checksum should be zero.

MAC80 Environment Control File Description

The MAC80 Environment Control file is read from logical device 23 during the initialization phase of cross-assembler execution. The file is referenced as FOR23.DAT in the DEC-20 system. Control commands are used to specify charactersitics of the source input file and to select available options on the type and extent of assembler output files. In addition to the control commands read from this file, control commands may be imbedded in the source file for dynamic respecification.

Selectable options include the following:

- Object file format
- Assembler internals dump selection
- Listing page ejection
- Input device selection
- Input source format definition
- Macro definition listing
- Output device selection
- Output suppression

For a more detailed description of function and use for the control commands available with MAC80

---

* See references, Page 33 .

please reference the External Reference Specification (3)* prepared by Intel, which has been filed with the Computing and Information Science Department.

MAC80 Assembler Error Processing Description

Errors detected in the source code during MAC80 execution are flagged with a single letter error indication on the output listing. If multiple errors exist on the same line of source code, only the first error is indicated. The assembler performs fundamental recovery when errors are encountered, by replacing defective expressions with one or more bytes of zeros. A complete list of error codes can be found in the External Reference Specification (3)* prepared by Intel, which has been filed with the Computing and Information Science Department.

---

* See references, Page 33 .

MICROPROCESSOR DEVELOPMENT EXECUTIVE FUNCTIONAL DESCRIPTION

The Microprocessor Development Executive was developed from another Intel 8080 based executive, RMUX, which was designed by Gary D. Crouse of the Bethlehem Steel Corporation Reasearch Department. RMUX (5)* is a real time multitasking executive intended for control applications. Many of the real time aspects of the executive were stripped out since they were not required for the microprocessor development system. The executive was transformed from a time driven system, where the primary system stimulus was a 50 millisecond clock interrupt, to an event driven system responding to interrupts from the system console device and the DECSYSTEM-20 input-output port. The executive acts as a monitor of user activities, and provides console communication between the user and the DEC-20, the executive debug facilities, and user developed programs. The executive includes logic to support a communications interface with the DECSYSTEM-20 computer. This interface permits two modes of communication. In the interactive mode, all key

entries from the users console are passed through the microprocessor and sent directly to the DEC-20. Since the DEC-20 supports half-duplex communication, the entered keys are echoed back to the microprocessor. Upon receipt of input from the DEC-20, the executive examines semaphores and, after verifying interactive mode, displays the received data on the system console device. The second mode of communications with the DECSYSTEM-20 is the download mode. This mode permits the user to request that microprocessor programs resident in the DEC-20 in object code format be loaded into the user area of the microprocessor memory for subsequent execution. Logic included in the executive issues a request to the DEC-20 for the user specified file, and loads the code into the microprocessor memory as it is received from the DEC-20.

A repertoire of system console commands is included in the executive program. These commands enhance the debug environment by permitting the user to modify and selectively execute portions of a downloaded program. Also included in the executive is a group of common service routine for performing utility functions. The common service routines can be referenced from user

programs. Descriptions of the system console commands and the common service routines available within the Microprocessor Development Executive are presented in the following sections. Source listings of the executive in both machine readable and printed form are available from the Computing and Information Science Department.

System Console Commands Description

System console commands may be entered from the system console whenever no solicited input requests (input requested from a user program) are outstanding. A single prompt character, ">", indicates that no solicited input is active. Input completion is signaled by a carriage return. Entered commands are directed to specific routines within the executive where parameters are verified for correctness. Erroneous input results in the display of a question mark at the console and the function is terminated. Command parameters are delimited by commas and blanks.

The command repertoire is presented in the following list. A document defining command syntax is available from the Computing and Information Science Department.

- Abort the currently executing user task
- Begin execution from specified address
- Begin execution with breakpoint installed
- Display memory contents
- Fill memory with hexidecimal data
- Move memory contents
- Single step execution
- Inspect and change memory location(s)
- Examine and change registers
- Download a program from the DEC-20

Common Service Routines Description

The Microprocessor Development Executive contains a group of user callable common service routines which provide utility functions. These routines offer frequently used conversion facilities as well as a software input-output interface to the system console. The common service routines may be referenced in user source programs through use of the assembler EQU statement to define the routine absolute address. Parameter descriptions and absolute

location addresses for the set of common service routines is available from the Computing and Information Science Department.

The types of utilities available are:

- Write contents of HL register pair to console
- Convert ASCII hexidecimal to binary value
- Convert BCD digits to binary
- Compare HL register pair to DE register pair
- Move a number of bytes in memory
- Write an ASCII string to the console with or without soliciting input
- Convert binary to ASCII character

A TYPICAL SCENERIO OF MICROPROCESSOR PROGRAM DEVELOPMENT

The following scenerio describes a normal procedure for developing microprocessor code using the Microprocessor Development System. The scenerio assumes that a design for the program has been completed and that a first draft of the assembly language code has been completed and is available for reference.

SOURCE CODE ENTRY AND ASSEMBLY

The user must enter the prepared source code into the DECSYSTEM-20 computer. For this purpose, the user logs on to the DEC-20 from any convenient terminal. It is usually advantageous to be connected to a 1200 baud port for this step of the development. After logging on to the DEC-20, the user envokes the editor for the purpose of creating a file containing the source code. The suggested file name for the source code is FOR20.DAT . This name may be subsequently changed but initially, FOR20.DAT will be more convenient since it

is the default input file referenced by the MAC80 Cross-assembler. After the source code has been entered, the file should be saved by specifying the unnumbered option, EU, when exiting the editor. After file creation is complete, the user envokes the MAC80 assembler by issuing a "RUN MAC80X" command. The cross-assembler will respond with the MAC80 version banner and a "BEGIN" signal. When assembly is complete, the number of assembly errors and "EXIT" prompt is presented at the terminal. If errors are present, the source code should be edited and reassembled, until all source errors have been eliminated. Once the assembly is error-free, a hard copy of the assembly listing, file FOB22.DAT should be printed. At this point, the user has completed initial object file preparation, and may log off from the terminal and proceed to the Intel iSBC 544 system for program testing and debug.

LOGGING ON THE DECSYSTEM-20 FROM THE ISBC 544 COMPUTER

Once an error free object file has been prepared on the DECSYSTEM-20, the user must move to the iSBC 544 computer in order to exercise the microprocessor program. The Microprocessor Development Executive, resident in PROM memory in the iSBC 544, goes through an initialization process following power-up of the computer. The system console and the acoustic modem must be attached to the iSBC 544 on serial input/output ports 0 and 1, respectively. A message acknowledging system initailization will be presented on the system console following initialization. The user must now dial up to the DEC-20 system in order for the communications link to be established. After verifying the carrier tone is present in the telephone headset, the user should place the headset into the modem cradle. Immediately after placing the phone in the modem, the user must alter the executive communications mode to interactive. This is accomplished by entering a "CNTL X" from the system console. Once interactive mode is established, a carriage return entry initiates a transmission to the

DEC-20 which results in the log-on prompt to be returned. The log-on procedure at this point is identical to that at any terminal, and successful completion of the log-on sequence establishes the communications session. All interactive DEC-20 commands are supported from the system console when in this mode.

LOADING THE OBJECT PROGRAM INTO THE ISBC 544 COMPUTER

Once the communications session with the DECSYSTEM-20 has been established, the user may exit the interactive mode again by entering a "CNTL X" from the system console. This entering and leaving the interactive mode may be performed any number of times during a communications session. Upon leaving the interactive mode, normal executive console commands are again available, including the LOAD command which performs the downloading of microprocessor object code. By entering the command, "L,FOR21" , the user will initiate the program download from the object file FOR21.DAT resident on the DEC-20 and cause the program

to be stored into the users area of random access memory. Completion of the program download is signaled at the system console by the "DOWNLOAD COMPLETE" message.

EXECUTING AND DEBUGING THE DOWNLOADED PROGRAM

After downloading the program into the users random access memory, the program may be inspected by use of system console commands. After verifying that the program appears correct, the "GO" command may be envoked to transfer control to the downloaded program. Breakpoints may be specified along with the "GO" command to trace program execution and the execution may be performed in single instruction steps with register inspection and memory inspection available after each step. If an instruction is detected to be in error, the code may be modified in memory and reexecuted. Modifications to the program should be noted on the hard copy of the program listing to permit subsequent source code correction on the DEC-20.

REESTABLISHMENT OF INTERACTIVE MODE

Once all detectable errors have been identified, the user may return to the interactive mode and make corrections to the source assembly language file. If the corrections are minor in nature and the user wishes to edit and recompile the source file via the Microprocessor Development System communications interface, this may be done. If extensive errors are identified and the user wishes to terminate the interactive session, this is also possible. The user must reenter the interactive mode via "CNTL X" entry. At this point the logoff or editor facilities are available, and the user may continue with the development procedure.

REFERENCES

(1) "Lehigh University Computing Center DECSYSTEM-20 User's Guide", Lehigh University, Bethlehem, Pa, 1979

(2) iSBC 544 Intelligent Communications Controller Board Hardware Reference Manual, Intel Corporation, Santa Clara, Ca, 1978

(3) External Reference Specification, 8080 Macro Assembler MAC80, Version 2, Intel Corporation, Santa Clara, Ca, 1974

(4) Intel 8080-8085 Assembly Language Manual, Intel Corporation, Santa Clara, Ca, 1974

(5) Crouse, Gary, "RMUX - Real Time Multitasking Executive Functional Description (Preliminary)", Internal Bethlehem Steel Corporation Research Department Document, Bethlehem, Pa, 1980

(6) Bethlehem Steel Corporation Research Department, Bethlehem, Pa, Numerous Discussions with Gary D. Crouse, Research Engineer, September, 1980 through April, 1981

V I T A

Charles Hagel Kercsmar was born in Fountain Hill, Pennsylvania on November 30, 1943 of parents Louis Kercsmar and Alyce Caroline Kercsmar nee Hagel. After serving in the armed forces, Mr. Kercsmar attended several institutions as a part-time student in pursuit of an undergraduate degree while remaining fully employed. These institutions include Lehigh University, the University of Vermont, Delaware Valley College, and Thomas A. Edison College. Mr. Kercsmar received a Bachelor of Science degree in Business Administration from Thomas A. Edison College in 1978.

Mr. Kercsmar is presently employed with the Bethlehem Steel Corporation as a Research Engineer. Prior to joining Bethlehem Steel, Mr. Kercsmar was a Staff Programmer with IBM Corporation Federal Systems Division.

Mr. Kercsmar is married to Kathleen A. Kercsmar and together they have three children, a daughter Melissa, and two sons, Michael and Geoffrey.