GENERAL SPECIFICATIONS

483

BIT INCORPORATED, 5 STRATHMORE ROAD, NATICK, MASSACHUSETTS

The BIT 483 Computer General Specifications
bulletin was compiled and written by
BIT, Incorporated.

Document No. 48302

Printed in U. S. A.
February, 1970

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

The BIT 483 Computer is a high-speed, general purpose, variable word length digital computer featuring an expandable memory, simultaneous input/output, and an extensive system of interrupts. The organization and built-in growth potential of the 483, implied in Figure 1, make it particularly well-suited for numerous business, scientific and engineering applications. Some of the most significant uses are:

A. Numerical Control
B. Data Communications
C. Instrumentation
D. Process Control
E. Data Acquisition
F. Education
G. Time Sharing
H. Business

GENERAL CHARACTERISTICS

The following list contains important characteristics pertinent to the 483.

A.	Memory Capacity		Variable from 1024 bytes to 65,536 bytes
B.	Memory Cycle Time		980 nanoseconds
C.	Word Size		Variable, multiples of 8-bit bytes
D.	Page Size		256 bytes
E.	Arithmetic		Binary (signed) Binary (unsigned) Decimal (signed)
		Name	Capacity
F.	Standard Data Channels (buffered)	C	1 teletypewriter
		G	32 peripheral devices, when only Channel G is used; 16 if both Channel A and Channel G are used
	Optional Data Channel (buffered)	A	16 peripheral devices
G.	Maximum Output Data Rate		8.16 million bits/second
	Maximum Input Data Rate		6 million bits/second
H.	Channel Modes (under program control)		Overlapped (cycle stealing) Non-overlapped
I.	Teletype Modes (under program control)		Buffered, non-interrupt mode Buffered, interrupt mode
J.	Priority Interrupts		8 basic (can be increased in groups of 8 up to 32)
K.	Sense Switches		9 sense switches
L.	Memory Protection		Contents of memory protected during power interruption by power failure interrupt
M.	Primary Power		115/230V, ±10%, 50/60 HZ
N.	Power Consumption		500 watts
O.	Weight		Approximately 50 pounds

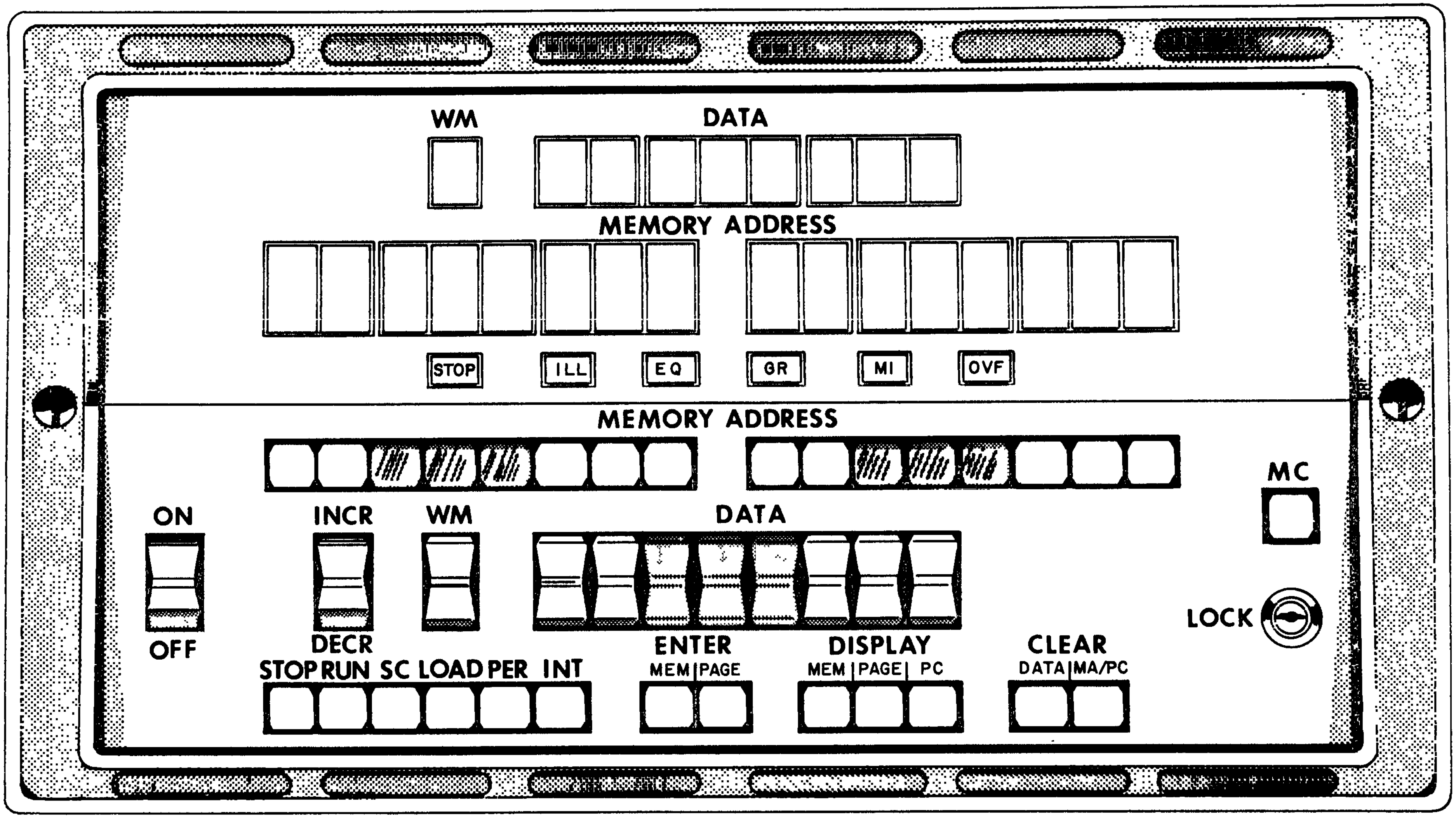


Figure 1. BIT Model 483 Computer

P.	Dimensions	10½ inches high x 19½ inches wide x 23 inches deep
Q.	Environmental Temperature	0 to 50°C
R.	Relative Humidity	95% maximum
S.	Mounting Capabilities	Standard 19 inch rack, pedestal, or table top
T.	Circuit Technology	Medium scale integration

EQUIPMENT ORGANIZATION

The 483 is a solid-state, stored program, digital data processor and computer constructed on the total modularity principle as illustrated in Figure 2. This feature enables the 483 to be easily organized and expanded by use of standard modular elements to satisfy the variety of needs in computer installation.

The operation of the 483 is controlled by an internally stored program located in the memory. Existing programs for the 483 are not affected by memory expansion. There may be a number of modules comprising a specific 483 system depending upon the particular application. Thus, a variety of equipment configurations can be organized. An illustration of some of the peripheral equipment which can be used with the 483 is shown in Figure 2.

FUNCTIONAL CHARACTERISTICS

The following paragraphs present brief descriptions of the major functional characteristics of the BIT 483 computer.

INTERNALLY STORED PROGRAM

Operation of the 483 is controlled by an internally stored program located in memory. A program consists of the series of instructions and related operational data and constraints required for performing a particular job.

FULLY SHARED MEMORY

The Central Processor and I/O Network have direct access to the totally shared core memory. Thus, the computing and I/O elements can directly communicate with all data.

INSTRUCTION REPERTOIRE

The instruction repertoire provided for the 483 includes instructions for signed decimal and signed or unsigned binary arithmetic, character and field operations, logical and control functions, varied data manipulations, conditional jump operations, and subroutine control. A tabulation of the basic instruction repertoire is given in Table 2.

VARIABLE WORD LENGTH

Variable word length capability in the 483 Computer enables the user to tailor the word size to the accuracy requirements of a problem. Multiple precision arithmetic, involving subroutines is never required. This conserves memory space.

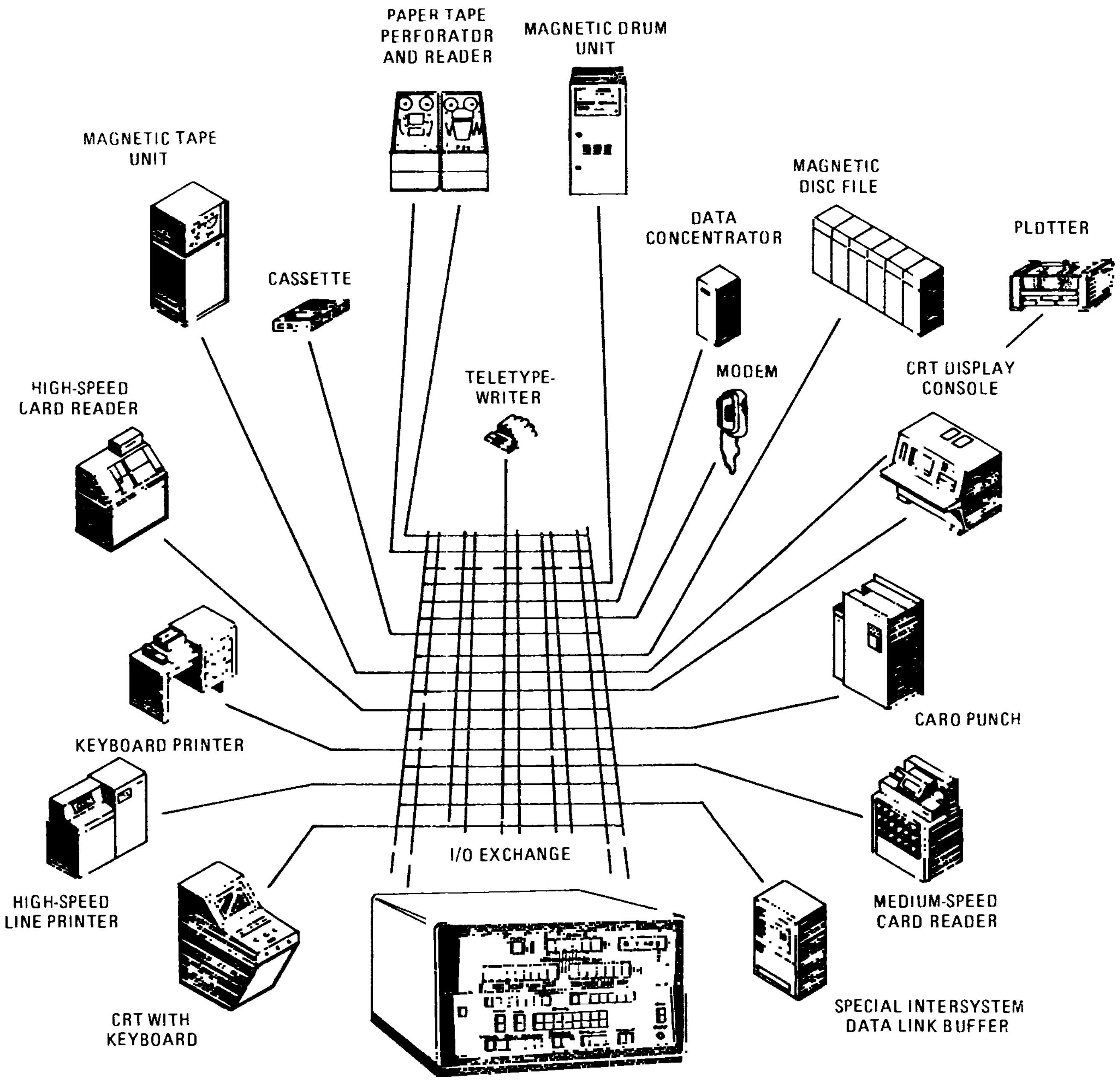


Figure 2. Equipment Configurations Possible with BIT 483 Computer.

It is equally easy to do 8-bit, 16-bit, or even 2,048-bit arithmetic with the 483 processor. In either case, only a single instruction is required.

Each standard arithmetic or logical instruction is bi-directional, in addition to being variable word length. In other words, either of the possibilities:

$$\text{(Storage)} + \text{(A)} \rightarrow \text{A}$$
$$\text{(Storage)} + \text{(A)} \rightarrow \text{Storage}$$

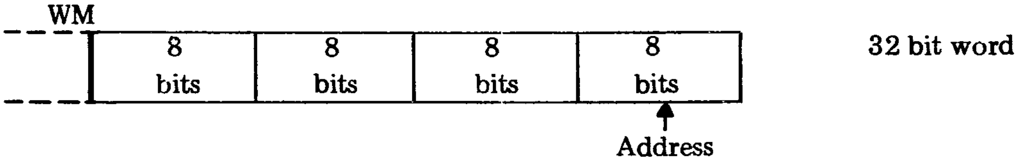
are selectable by means of a control bit in the instructions, simplifying programming considerably.

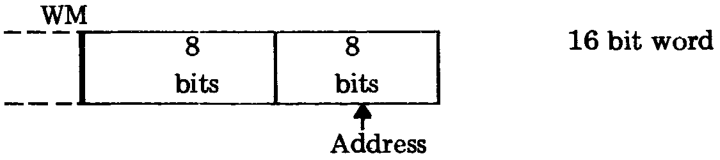
There are four indicator flops set during every arithmetic and logical instruction: EQ (equal to zero), GR (greater than zero), MI (minus indicator), OVF (overflow). These indicators may be interrogated as a condition state for subsequent jump instructions.

The examples in Figure 3 illustrate only a fraction of the power inherent in the 483 variable word length capability. In each example, only a single instruction is required.

WORD SIZE

A word in the 483 processor is any integral number of bytes as shown below.





The word is addressed at its low order byte and is word marked at its high order end.

Both the accumulator and words in core memory are variable by program control. This flexibility permits the programmer to effect maximum efficiency in his use of core memory, and simplifies programming in many situations. Software provided by BIT is structured to allow the user to take maximum advantage of mixed word length processing at his discretion.

MEMORY ADDRESSING

Memory locations are addressed by unsigned positive binary numbers. For addressing purposes the memory is divided into blocks of 256 bytes called pages. Each 2-byte instruction can directly address 512 bytes, in either the page containing the next instruction or a page previously designated by the program as the data page. Indirect addressing capability also allows the user to address indirectly any location within either the program counter page or the page at which the page register is set. The 4-byte instructions can address any location in any page.

1. Unsigned Binary Subtract two 24 bit numbers (result in accumulator)
Instruction: BSA X

INDICATORS

				EQ	GR	MI	OVF
ACC	1 0 0 1 1 1 0 0	0 1 1 1 1 0 0 1	0 0 0 0 1 0 1 0				
X	0 1 1 1 1 1 1 0	1 0 0 0 0 1 0 0	0 1 0 1 0 0 1 1				
ACC RESULT	0 0 0 1 1 1 0 1	1 1 1 1 0 1 0 0	1 0 1 1 0 1 1 1	0	1	0	0

2. Signed Binary Subtract two 16 bit numbers (result in accumulator)
Instruction: BSA Y

			EQ	GR	MI	OVF
ACC	1 0 0 1 1 1 0 0	0 1 1 1 1 0 0 1				
Y	0 0 0 0 1 1 1 0	1 0 0 0 0 1 0 0				
ACC RESULT	1 0 0 0 1 1 0 1	1 1 1 1 0 1 0 1	0	1	1	0

3. Signed Binary Add two 24 bit numbers (result in memory)
Instruction: BAM X

				EQ	GR	MI	OVF
	1 1 0 0 1 1 0 0	1 1 1 1 0 0 0 0	1 0 1 0 1 0 1 0				
ACC	1 1 1 1 0 0 0 0	1 1 0 0 1 1 0 0	0 1 0 1 0 1 0 1				
X RESULT	1 0 1 1 1 1 0 1	1 0 1 1 1 1 0 0	1 1 1 1 1 1 1 1	0	1	1	0

4. Decimal Add two 6 digit decimal numbers (result in accumulator)
Instruction: DAA X

							EQ	GR	MI	OVF
ACC	6	1	9	.	3	2				
X	2	3	1	.	7	8				
ACC RESULT	8	5	1	.	1	0	0	1	*	*

5. Decimal Subtract two 4 digit decimal numbers (result in accumulator)
Instruction: DSA Y

					EQ	GR	MI	OVF
ACC	4	3	2	1				
Y	9	8	7	6				
ACC RESULT	5	5	5	(–) 5	0	0	*	*

6. Exclusive OR two 16 bit numbers (result in memory)
Instruction: ORM X

			EQ	GR	MI	OVF
X	1 1 0 0 1 1 0 0	1 1 1 1 0 0 0 0				
ACC	1 1 1 1 0 0 0 0	1 1 0 0 1 1 0 0				
X RESULT	0 0 1 1 1 1 0 0	0 0 1 1 1 1 0 0	0	1	*	*

7. Logical AND two 8 bit numbers (result in accumulator)
Instruction: ANA Y

		EQ	GR	MI	OVF
ACC	1 1 0 0 1 1 0 0				
Y	1 1 1 1 0 0 0 0				
ACC RESULT	1 1 0 0 0 0 0 0	0	1	*	*

*remains unchanged

Figure 3. Variable Word Length Operations.

TYPES OF DATA WORDS

The data words consist of any multiple of 8-bit bytes where the most significant byte has a word mark bit. These bytes of information are used in the computer in the form of four different types of data words:

A. Binary
B. Purely numeric
C. Purely alphabetic
D. Alphanumeric

TIMING

The time required for one memory cycle, i.e., transferring a byte between the Central Processor and the memory, is 980 nanoseconds. The time to fetch an instruction is measured in memory cycles. Instruction execution times are discussed in the following paragraph.

EXECUTION TIMES

Instruction execution times vary with the instruction type, number of addressed bytes, the extent of use of indirect addressing, etc. For example, the average execution time for a 2-byte jump instruction is .98 microseconds; for a binary add, 2.3 microseconds; and for a binary subtract, 2.3 microseconds.

PRIORITY INTERRUPT CAPABILITY

The interrupt subsystem of the 483 consists of an expandable priority chain. Eight priority levels are provided as standard equipment on each 483.

The four highest level interrupts are internally generated by the 483. The rest are available for use by the customer. Additional interrupt levels may be provided as optional equipment (in groups of 8) up to a maximum of 32 levels.

The standard level assignments are summarized below:

Level 1	Power failure
Level 2	Program
Level 3	Teletype (C Channel)
Level 4	Front panel interrupt button
Level 5,	
6, 7, 8	Open

The interrupt system allows interrupts of interrupts to an indefinite number of levels. Upon each entrance to an interrupt, the program counter and indicator flops are stored, by hardware action, for subsequent restoration upon completion of the interrupt subroutine. An interrupt at a given priority level can interrupt all lower priority interrupts.

AUTOMATIC POWER FAILURE INTERRUPT

The automatic power failure interrupt, listed in "Priority Interrupt Capability," is provided in order to allow the 483 sufficient time to react when the line voltage drops below a prescribed level. When the processor detects a power failure condition, the program counter and all indicators are automatically stored by hardware action. The program is then notified by interrupt that the system is experiencing a power failure so that peripheral devices may be smoothly cycled down. When line voltage is restored, the 483 can resume operation at the point in the program where the interrupt occurred.

DATA TRANSFER

The maximum output data transfer rate of the 483 is 1.02 million bytes per second, or 8.16 million information bits per second.

The maximum input data transfer rate is 0.75 million bytes per second, or 6 million information bits per second.

A unique feature of the 483 processor is that the data transfer can be commanded to be either overlapped (simultaneous with computation) or non-overlapped (machine hangs until completion of transfer). This choice is under program control, by means of a bit in the peripheral transfer instruction. Data transfers are fully buffered by hardware in the 483 Direct Memory Access (DMA) data channels.

Since the 483 is a variable word length machine, the data transfer instruction can transmit either a single byte or an entire block of data bytes of any length. A second feature of the transfer is that it can be commanded to start transmission from any location in core memory.

A number of methods can be employed as a means of terminating a block transfer, depending upon the requirements of the I/O device. A very simple technique, possible because of the flexible 483 word structure, is that of having a word mark end the transfer. Instructions for implementing this are provided in BIT I/O literature.

For an output data transfer, the execution time is .98 microseconds + .98 n microseconds. (One memory cycle is required to set up the data channel, and one cycle is required for each of n bytes transmitted by the instruction.)

For an input data transfer, the execution time is .98 microseconds + 1.33 n microseconds. (One memory cycle is required to set up the data channel and 1.33 microseconds are required for each data byte entered into core memory.) When data is entered into core memory from an I/O device, the word mark structure in memory is preserved.

For the convenience of users who are designing interfaces, BIT offers a series of dedicated logic interface adapter cards containing device selection and synchronizing logic. Use of these cards greatly simplifies the task of interface design. As the user is well aware, every computer has timing functions which must be implemented in the design of an interface. With these cards, the user is relieved of the task of generating these signals. (The reader is referred to BIT manual "Description of Input/Output.")

483 SOFTWARE

Software for the 483 is versatile and in-depth, proven in a broad range of applications with many BIT computer installations.

The software provided with the 483 ranges from compiler level to assembler level, supported by extensive utility systems and mathematical aids.

ASSEMBLERS

Five assemblers are provided for use with the 483 Computer, varying in size, sophistication and ease of use. The assemblers permit the programmer to code instructions in a symbolic language which is more convenient than the binary numbers which actually operate the machine.

A. ABIT-1: This symbolic assembly language system is designed for 483 units having a 1024 or 2048 byte memory capacity.

B. ABIT-4: This symbolic assembly language system is designed for 483 units having a 4096 byte memory or larger.

C. MABIT: The MABIT is a modified ABIT-4 which includes 17 interpretive floating-point function single-statement calls, including HTAN, ARCTAN, TRIG, LOG, e^x, and others.

D. MACROBIT: A macroassembler for use with 8K memory or larger, with a novel form for macro calls and a syntax using the full ASCII character set which permits considerable economy of expression. Macrobit can be thought of as a powerful string processor followed by a very simple assembler.

The string processor features include: "if" and "repeat" statements, definition and redefinition of string macros (variables which expand to strings), and a form for macro calls in which there are no special delimiters setting of a macro's arguments.

The simple assembler which operates on the string generated by the string processor has the features normally found in an assembler. These include: complete symbol table control for the programmer allowing definition, redefinition, purge, read-in, type out, and punch out, and extensive address arithmetic, good error diagnostics, and convenient option selection via the console switches.

The macroassembler is able to use the same storage allocation algorithm described in IL, relieving the programmer of the task of fitting small segments of program into page-size pieces.

E. IL: Is an advanced programming language for use with 8K memories or larger, pioneered by BIT for minicomputer applications. It permits the user to program at a level of detail intermediate between assembly language and conventional higher level languages. He can, for example, write assignment and conditional statements as in higher level languages, yet has extensive control over the details of the code generated and full use of all of the features of the machine. The language is more compact to use than an assembler in the sense that a statement in it may encompass a number of operations. As an example, the statement

$$A \leftarrow B + C - D \& A$$

in IL effects the following:

B is placed in the accumulator. C is added to it. D is subtracted from it. The result of this is then logically ANDed with A. The result is then placed in A.

The programmer, using IL, has all the advantages associated with assembly language and at the same time much of the conciseness and readability of higher level languages.

In addition, a unique storage allocation feature is provided. The programmer writes his program in segments, to each of which he can assign an absolute origin if desired; if he does not, the allocator makes an efficient assignment of these segments into the available core space. The result of an IL compilation is an absolute binary tape. This allocation process greatly simplifies the task of fitting programs efficiently into a core paged machine.

UTILITY SYSTEMS

The 483 software features several utility programs which are designed to facilitate programming and hardware operations. These features include a Text Editor, EDIT; an on-line debugging system, BOLD; automatic loaders, BINREAD and LOADER-T; and punch programs, BINPUNCH.

A. EDIT: EDIT is a text editor which permits source program modification such as corrections, additions, deletions, etc., in the assembly language.

B. SYTE: For processors with minimum memory sizes, not large enough to accommodate EDIT. SYTE permits source program modification in assembly language, corrections, additions and deletions.

C. BOLD: BOLD is an on-line debugging system used in debugging new programs. BOLD permits entry of break-points, subprogram entries and exits, manual data entry, memory data listing and psuedo interrupts. It also permits direct operator control of all phases of processing. BOLD uses 1.5K of core storage.

D. BINREAD and LOADER-T: These programs locate and load object programs and data into preselected memory areas. As programs are operating these loaders reside in memory ready for reuse whenever needed.

E. BINPUNCH: BINPUNCH may be used independently or as a segment of BOLD to copy selected memory data or programs onto punched paper tape. The use of BINPUNCH makes it possible to copy a modified operating program directly onto paper tape for reuse at a later time; thereby eliminating the need for reassembling a program when it must be corrected or modified.

FORTRAN AND MATHEMATICAL AIDS

The 483 software also features a FORTRAN compiler; a set of mathematical subroutines, MATH PACK, which perform floating-point arithmetic, and a system of automatically performing fifteen arithmetic keyboard calculations, called CALC.

A. FORTRAN: The FORTRAN compiler, designed for use with a 483 having an 8K memory or larger, includes every function of USA STANDARDS BASIC FORTRAN (which is approximately equivalent to an "E" level subset of full FORTRAN IV). It is a one-pass compiler which produces an object program. It greatly simplifies the problem of program preparation and enables users with little or no knowledge of the computer's organization to write programs by expressing problems in a mixture of mathematical statements and English words.

B. MATH PACK: The MATH PACK consists of a set of subroutines which perform floating-point arithmetic. The set includes add, subtract, multiply, divide, sine, cosine, tangent, square root, exponential and log.

C. CALC: CALC essentially converts the 483 into a rapid desk calculator capable of automatically performing fifteen arithmetic functions. The ASR-33 teletype keyboard is utilized for input and output.

HARDWARE ELEMENTS

The 483 consists of four major hardware elements:

A. I/O Network
B. Central Processor
C. Memory
D. Control Panel

I/O NETWORK

The 483 I/O Network consists of a number of data channels, each capable of operating in a Read-Write mode. The Central Processor can have a maximum of two such Read-Write channels, as shown in Figure 4. Each of these channels are capable of executing data transfers while the processor is simultaneously computing. A maximum of 32 peripheral devices and a teletype can be addressed via the I/O Network.

The teletype used with the 483 (unless otherwise specified) is a standard ASR-33. A tabulation of the Teletype Code, octal and binary, is given in Table 3 at the end of this bulletin. By program control, the teletype can be commanded to operate in either interrupt mode or non-interrupt mode.

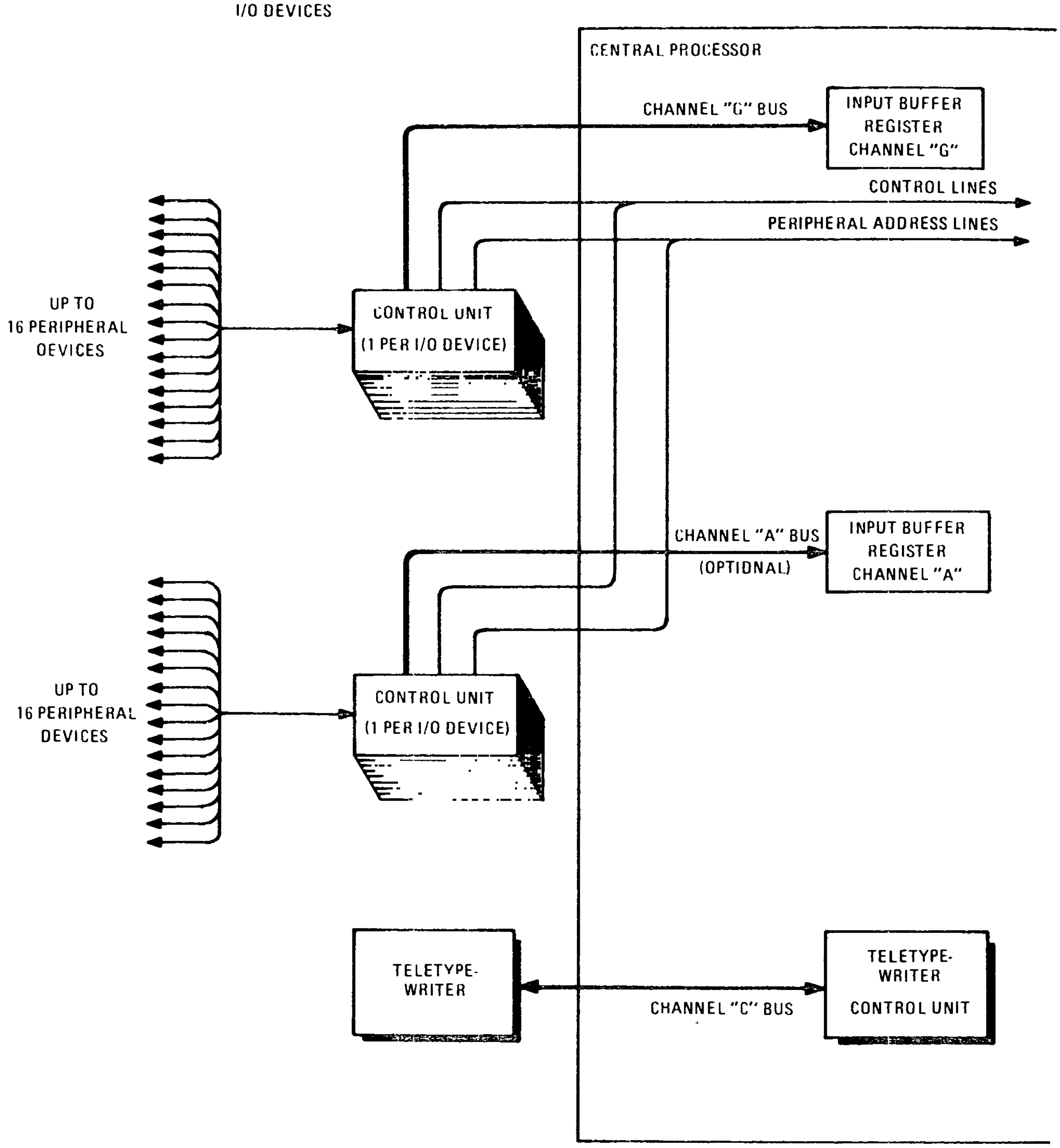

I/O DEVICES
CENTRAL PROCESSOR
CHANNEL "G" BUS
INPUT BUFFER REGISTER CHANNEL "G"
CONTROL LINES
PERIPHERAL ADDRESS LINES
UP TO 16 PERIPHERAL DEVICES
CONTROL UNIT (1 PER I/O DEVICE)
CHANNEL "A" BUS (OPTIONAL)
INPUT BUFFER REGISTER CHANNEL "A"
UP TO 16 PERIPHERAL DEVICES
CONTROL UNIT (1 PER I/O DEVICE)
TELETYPE-WRITER
CHANNEL "C" BUS
TELETYPE-WRITER CONTROL UNIT

Figure 4. Block Diagram — I/O Network.

CENTRAL PROCESSOR

The Central Processor basically consists of: 1) a set of registers to store various pieces of transient information; 2) transfer buses for transferring information among the various registers or between the Central Processor registers and other parts of the system; 3) control logic to execute the basic cycle of fetching and decoding an instruction, address modification, etc.; and 4) control logic for executing the various individual instructions in the 483 repertoire. A block diagram of the Central Processor is given in Figure 5.

The processor has a single-address instruction capability and access to all memory locations in the main core memory. The Central Processor is controlled by the program stored in memory and is capable of processing data and performing arithmetic and logical operations. The logic is implemented with extensive Medium Scale Integration (MSI) techniques and Transistor-Transistor Logic (TTL) elements.

MEMORY

The magnetic core memory is the primary storage facility for the 483. It provides rapid, random access data and instruction storage for both the Central Processor and the I/O Network. The basic memory consists of 18 planes, 16 of which are used to store 8-bit characters called bytes and 2 planes which store word mark bits. A pair of additional planes, an optional feature, is available for use as an odd parity bit for each byte. The minimum memory size available is 1024 bytes, expandable times 2, up to 65,536 bytes. A simplified diagram of the memory core layout is given in Figure 6.

CONTROL PANEL

The control panel, shown in Figure 1, is divided into two main sections: indicator lamps in the top section and switches below. The indicators can display the status of the 483, i.e., stop, illegal instruction, etc., as well as any significant register. In addition to their normal functions, the switches have the capability of accessing and changing the contents of any memory location. Identification and function of panel switches and indicators are given in Table 1.

Table 1. Identification of Switches and Indicators

Name	Function
ON/OFF	Turns 115 VAC primary voltage "on" and "off".
STOP (Momentary Contact Switch)	Stop switch. When pressed, program execution stops. Stop indicator lamp lights.
RUN (Momentary Contact Switch)	Run switch. When pressed, starts program execution at the address specified in the program counter.
SC (Momentary Contact Switch)	Single cycle switch. When pressed, executes a single instruction.

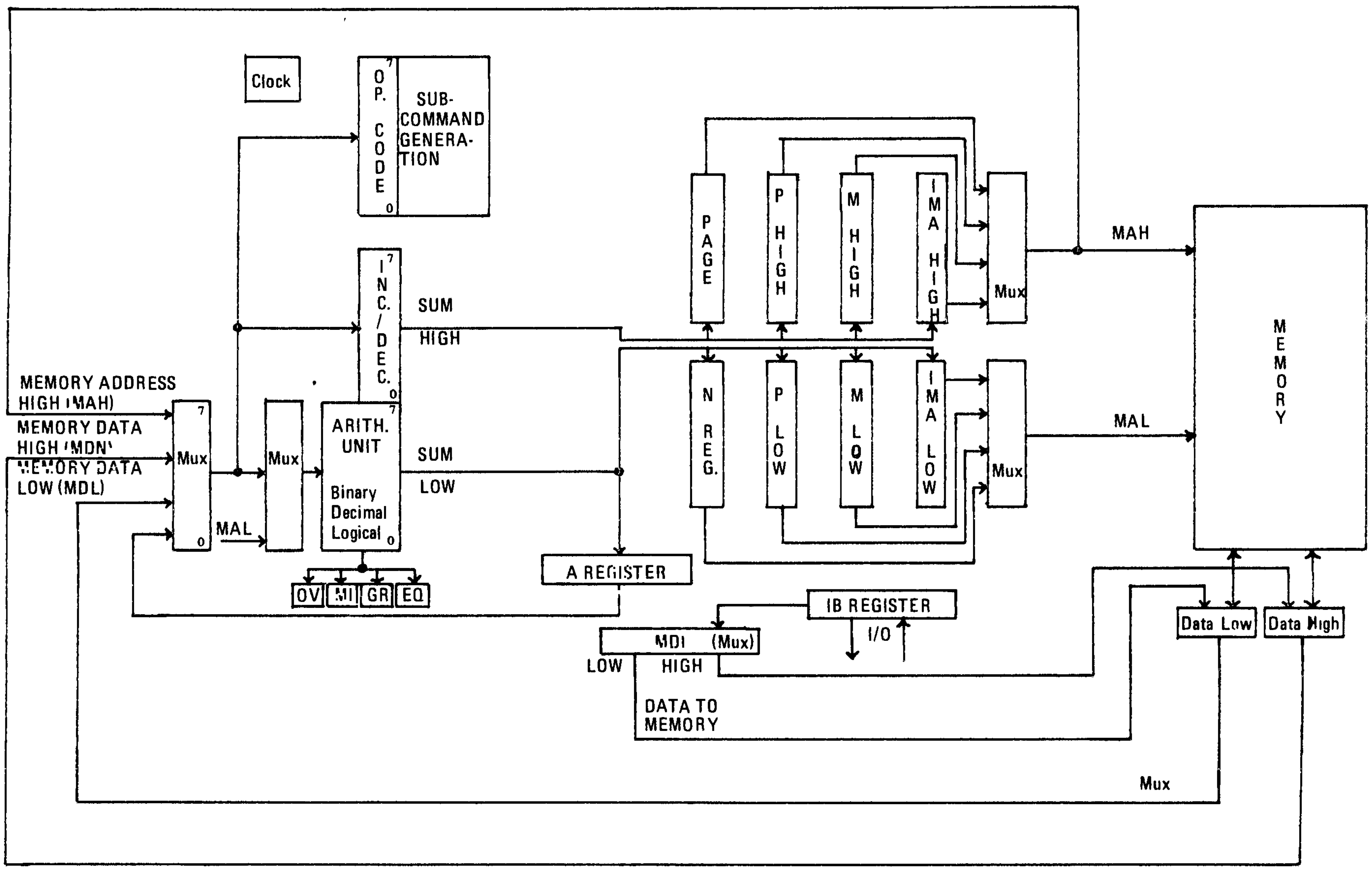


Figure 5. Block Diagram — Central Processor

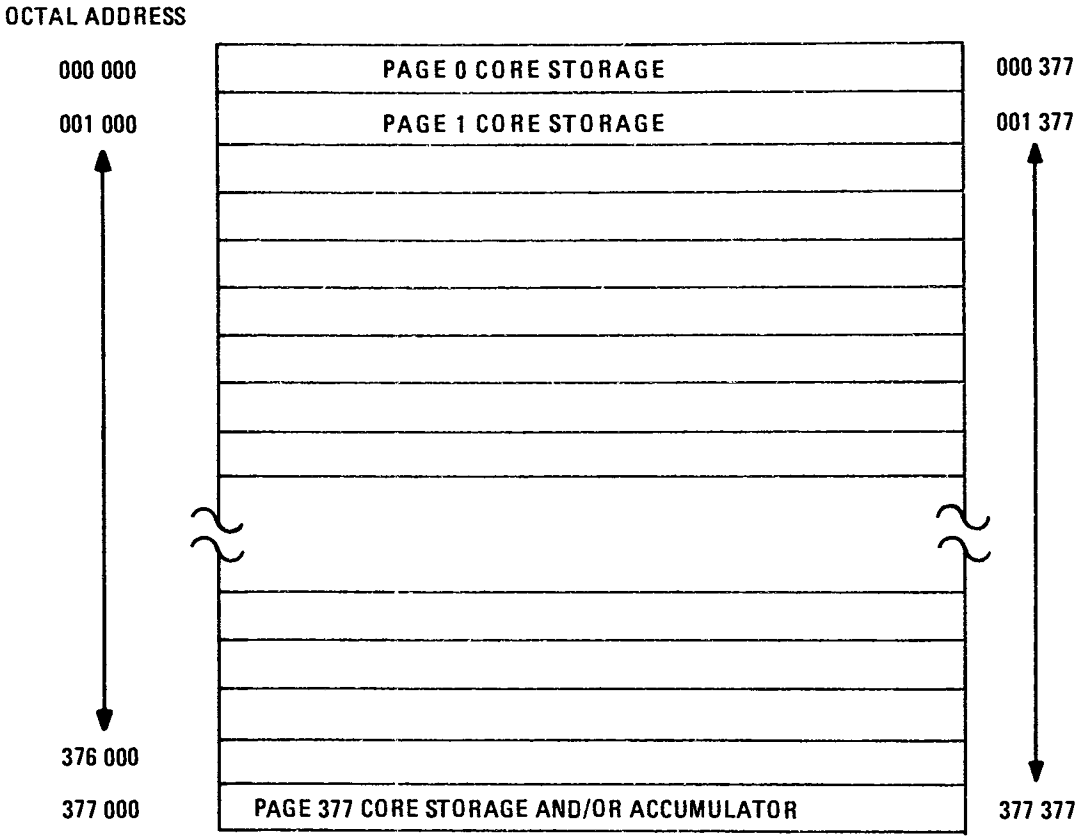


Figure 6. Memory Core Layout

Table 1 (continued)

Name	Function
LOAD (Momentary Contact Switch)	Load switch. When pressed, starts to read paper tape into memory from teletype reader; stopped automatically by blank character on tape.
PER (Momentary Contact Switch)	Peripheral switch. When pressed, resets all peripheral lines to zero without affecting the 483 processor registers.
INT (Momentary Contact Switch)	Interrupt switch. When pressed, interrupts processor operation.
ENTER MEM (Momentary Contact Switch)	Enter memory switch. When pressed, data displayed on DATA indicators are actually entered into the memory at the address shown on the MEMORY ADDRESS indicators.
ENTER PAGE (Momentary Contact Switch)	Enter page switch. When pressed, data displayed on DATA indicators are actually entered into the page register.
DISPLAY MEM (Momentary Contact Switch)	Display memory switch. When pressed, displays on the DATA indicators the information stored at the address displayed on the MEMORY ADDRESS indicators. The contents of sequential memory locations are displayed each time the switch is pressed.
DISPLAY PAGE (Momentary Contact Switch)	Display page switch. When pressed, displays contents of page register on DATA indicators.
DISPLAY PC (Momentary Contact Switch)	Display program counter switch. When pressed, displays location of next instruction in program counter on MEMORY ADDRESS indicators.
CLEAR DATA (Momentary Contact Switch)	Clear data switch. When pressed, clears data appearing on DATA display indicators.
CLEAR MA/PC (Momentary Contact Switch)	Clear memory address/program counter switch. When pressed, clears the contents of the memory address register, the program counter, and the MEMORY ADDRESS indicators.
LOCK	A manual lock and key is provided to lock the control panel switches in order to prevent unintentional "pressing" of switches. When the key is horizontal, the panel is locked; vertical, open.
MC (Momentary Contact Switch)	Master clear switch. When pressed, resets all major registers and peripheral lines to zero.

Table 1 (continued)

Name	Function
INCR/DECR	Increment/Decrement switch. Causes the main memory address register to be incremented or decremented (depending on whether the switch is in the INCR or DECR position) when either the ENTER MEM or DISPLAY MEM switch is pressed.
WM	Word mark switch. When pressed, enters the word mark bit (9th bit of a byte) into the DATA indicators.
DATA Switches	A series of DATA switches provides the capability of manually entering data into the memory. The switches are colored in groups to facilitate reading them in the octal system.
SENSE Switches	Nine SENSE switches are available for use on the control panel. Any permutation of these switches can be tested by the program to determine if specialized processing is necessary. It is possible to set and reset any code when the CPU is actively engaged in processing other requirements.
	INDICATORS
STOP Indicator	Lights when the STOP switch is pressed, when the program halts, or when the computer tries to execute an illegal instruction.
ILL Indicator	Illegal operation indicator. Lights when the processor halts when attempting to execute an illegal operation code.
EQ Indicator	Equal to zero indicator. Lights when the result of a binary logical or decimal operation is zero, or it and the GR indicator are both set to indicate word size has been exceeded, i.e., when there is a carry out of the highest order byte of the results.
GR Indicator	Greater than zero indicator. Lights when the result of a decimal operation is greater than zero, or the result of a binary or logical operation is non-zero, or it and the EQ indicator are both set to indicate word size has been exceeded (as per the preceding explanation).
MI Indicator	Minus indicator. Lights when the highest order bit of the result of a binary operation is a binary one.

Table 1 (continued)

Name	Function
OVF Indicator	Overflow indicator. Lights after a binary operation when either a "carry out" of the highest order bit of the result without an accompanying "carry into" it occurs, or when a "carry into" the highest order bit occurs without an accompanying "carry out."
WM Indicator	Word mark indicator. Lights when 9th word bit is set.
DATA Indicators	Lights when DATA switches are pressed. During operation indicate data entering or being accessed from memory.
MEMORY ADDRESS Indicators	Light when MEMORY ADDRESS Switches are pressed. During operation indicate the current memory address.

Table 2. Basic 483 Instructions

Instruction	Execute Timing (cycles)	Mnemonic	Binary Code	Octal Code	Functional Description
		Jump Instructions			
Jump return from interrupt	3.0	*JRI	00100000	040	Jump return from interrupt
Unconditional jump	3.0	*JMP	00100001	041	Unconditional jump
Jump if EQ	3.0	*JEQ	00100010	042	Jump if equal flop is set
Jump if $\overline{\text{EQ}}$	3.0	*NEQ	00100011	043	Jump if equal flop is not set
Jump if GR	3.0	*JGR	00100100	044	Jump if greater flop is set
Jump if $\overline{\text{GR}}$	3.0	*NGR	00100101	045	Jump if greater flop is not set
Jump if GR & EQ	3.0	*JGE	00100110	046	Jump if greater and equal flops are both set
Jump if $\overline{\text{GR \& EQ}}$	3.0	*NGE	00100111	047	Jump if greater or equal flops are not set
Jump on peripheral test	3.0	*JPT	00101000	050	Jump if positive response from peripheral equipment

NOTES:

*Fetch Timing = 2 memory cycles
Memory Cycle Time = 980 nanoseconds
Indirect Addressing = 1 cycle
n = the number of bytes in either the memory or accumulator operand, whichever is smaller
nm = the number of bytes in the memory word
na = the number of bytes in the accumulator word
S = time between data bytes from peripheral device

Table 2 (continued)

Instruction	Execute Timing (cycles)	Mnemonic	Binary Code	Octal Code	Functional Description
Jump to subroutine	5.6	*JSR	00101001	051	Jump to subroutine
Jump if MI	3.0	*JMI	00101010	052	Jump if minus flop is set
Jump if $\overline{\text{MI}}$	3.0	*NMI	00101011	053	Jump if minus flop is not set
Jump if OV	3.0	*JOV	00101101	054	Jump if overflow flop is set
Jump if $\overline{\text{OV}}$	3.0	*NOV	00101101	055	Jump if overflow flop is not set
Jump if EQ or GR	3.0	*EOG	00101110	056	Jump if equal or greater flop is set
Jump if $\overline{\text{EQ or GR}}$	3.0	*ENG	00101111	057	Jump if greater flop is not set and if equal flop is not set
		Peripheral Control Instruction			
Peripheral select		PCI	0011DDDD		Selects device DDDD
		Logical Instructions			
Exclusive or to accumulator	2.3 n	ORA	01000P0I	100	Exclusive or the contents of memory to the contents of the accumulator
Exclusive or to memory	2.3 n	ORM	01000P1I	102	Exclusive or the contents of the accumulator to the contents of memory
Logical and to accumulator	2.3 n	ANA	01001P0I	110	And the contents of memory to the contents of the accumulator

Table 2 (continued)

Instruction	Execute Timing (cycles)	Mnemonic	Binary Code	Octal Code	Functional Description
Logical and to memory	2.3 n	ANM	01001P1I	112	And the contents of the accumulator to the contents of memory
		Binary Instructions			
Binary add to accumulator	2.3 n	BAA	01010P0I	120	Binary add the contents of memory to the contents of the accumulator
Binary add to memory	2.3 n	BAM	01010P1I	122	Binary add the contents of the accumulated to the contents of memory
Binary subtract from accumulator	2.3 n	BSA	01011P0I	130	Binary subtract the contents of memory from the contents of the accumulator
Binary subtract from memory	2.3 n	BSM	01011P1I	132	Binary subtract the contents of the accumulator from the contents of memory
		Decimal Instructions			
Decimal add to accumulator	2.3 n	DAA	10100P0I	240	Decimal add the contents of memory to the contents of the accumulator
Decimal subtract from accumulator	2.3 n	DSA	10101P0I	250	Decimal subtract the contents of memory from the contents of the accumulator
		Page Register Instructions			
Change page register	1.0	CHP	01100P0I	140	Replace the contents of the page register with the contents of memory
Store page register	1.3	SPR	01100P1I	142	Save the contents of the page register in memory
Store sense switches	1.3	SSS	01101P1I	152	Save the sense switch settings in memory

Table 2 (continued)

Instruction	Execute Timing (cycles)	Mnemonic	Binary Code	Octal Code	Functional Description
Copy Instructions					
Move one word to accumulator	2.0 nm	C1A	01110P0I	160	Copy one word from memory to the accumulator
Move one word to memory	2.0 na	C1M	01110P1I	162	Copy one word from the accumulator to memory
Move two words to accumulator	4.0 nm	C2A	01111P0I	170	Copy two words from memory to the accumulator
Move two words to memory	4.0 na	C2M	01111P1I	172	Copy two words from the accumulator to memory
Word Mark Instructions					
Erase word mark	1.3	EWM	10000P0I	200	Erase the word mark bit in the specified memory location
Set word mark	1.3	SWM	10000P1I	202	Set the word mark bit in the specified memory location
Jump on word mark	3.0	*JWM	10001P0I	210	Jump if the word mark bit is set in the specified memory location
Two-Byte Jump Instructions					
Two-byte jump	1.0	PGJ	10010P0I	220	Change program counter
Jump & halt	1.0	HLT	10011P0I	230	Change program counter and halt

Table 2 (continued)

Instruction	Execute Timing (cycles)	Mnemonic	Binary Code	Octal Code	Functional Description
Set interrupt & jump	1.0	SIF	10010P1I	222	Change program counter and set interrupt flop
Teletype Non-Interrupt Mode					
Transfer from TTY paper tape reader non-interrupt mode	100 milliseconds	RPT	00011010	032	Initiates a transfer from the paper tape reader on the TTY
Transfer from TTY keyboard non-interrupt mode	100 milliseconds	KEY	00011000	030	Initiates a transfer from the keyboard on the TTY
Transfer to TTY punch/keyboard non-interrupt mode	100 milliseconds	PRT	00011001	031	Initiates a transfer to the punch/printer on the TTY
Peripheral Transfer Instructions					
Non-overlapped input peripheral transfer Channel G	1+1.3N+S	PTG	00011100	034	Initiates a non-overlapped transfer to a previously selected device on I/O Channel G
Non-overlapped output peripheral transfer Channel G	1+1.0N+S	PTG	00011100	034	Initiates a non-overlapped transfer to a previously selected device on I/O Channel G
Non-overlapped input peripheral transfer Channel A	1+1.3N+S	PTA	00010000	020	Initiates a non-overlapped transfer to a previously selected device on I/O Channel A
Non-overlapped output peripheral transfer Channel A	1+1.0N+S	PTA	00010000	020	Initiates a non-overlapped transfer to a previously selected device on I/O Channel A

Table 2 (continued)

Instruction	Execute Timing (cycles)	Mnemonic	Binary Code	Octal Code	Functional Description
Overlapped input peripheral transfer Channel G	1+1.3N	OTG	00011101	035	Initiates an overlapped transfer to a previously selected device on I/O Channel G
Overlapped output peripheral transfer Channel G	1+1.0N	OTG	00011101	035	Initiates an overlapped transfer to a previously selected device on I/O Channel G
Overlapped input peripheral transfer Channel A	1+1.3N	OTA	00010001	021	Initiates an overlapped transfer to a previously selected device on I/O Channel A
Overlapped output peripheral transfer Channel A	1+1.0N	OTA	00010001	021	Initiates an overlapped transfer to a previously selected device on I/O Channel A

Table 3. The ASCII Teletype Code (Octal & Binary)

Character	Octal	Binary	Character	Octal	Binary
A	301	11000001	!	241	10100001
B	302	11000010	“	242	10100010
C	303	11000011	#	243	10100011
D	304	11000100	$	244	10100100
E	305	11000101	%	245	10100101
F	306	11000110	&	246	10100110
G	307	11000111	’	247	10100111
H	310	11001000	(	250	10101000
I	311	11001001	)	251	10101001
J	312	11001010	*	252	10101010
K	313	11001011	+	253	10101011
L	314	11001100	,	254	10101100
M	315	11001101	—	255	10101101
N	316	11001110	.	256	10101110
O	317	11001111	/	257	10101111
P	320	11010000	:	272	10111010
Q	321	11010001	;	273	10111011
R	322	11010010	<	274	10111100
S	323	11010011	=	275	10111101
T	324	11010100	>	276	10111110
U	325	11010101	?	277	10111111
V	326	11010110	@	300	11000000
W	327	11010111	[	333	11011011
X	330	11011000	\	334	11011100
Y	331	11011001	]	335	11011101
Z	332	11011010	↑	336	11011110
			←	337	11011111
0	260	10110000	Line/Feed	212	10001010
1	261	10110001	Carriage/Return	215	10001101
2	262	10110010	Space	240	10100000
3	263	10110011	Rub-Out	377	11111111
4	264	10110100			
5	265	10110101			
6	266	10110110			
7	267	10110111			
8	270	10111000			
9	271	10111001			