MONROBOT XI PROGRAM MANUAL

Copyright 1960
Monroe Calculating Machine Company, Inc.
Orange, New Jersey
Form MO-96 ED 1-64
Printed in U.S.A.

INTRODUCTION

The purpose of this manual is to acquaint the programmer and coder with the machine language of Monrobot Mark XI. The manual is written for those people who are experienced in the programming and coding of general purpose computers. It is assumed that most computer terms used here are familiar to the reader; terms that are specific to the Monrobot Mark XI are defined as they are introduced into the text.

No attempt has been made to provide either introductory comments about computers or methods of programming computers. For those people not experienced in programming, Monroe literature or some text on basic computer programming should be studied prior to reading and using this manual.

Other Monroe literature defines the specific techniques of numeric input and output, alphanumeric handling, multiplication and division, and program input/output.

The scope of this manual is believed sufficiently broad to enable people to program, code, and understand Monrobot Mark XI in a reasonably short period of time. Consequently, some of the more esoteric information about the computer has been omitted.

TABLE OF CONTENTS

GENERAL

Monrobot Mark XI is a small, general purpose, digital, transistorized computer. Its number system is straight binary. The program is internally stored along with data on a magnetic drum. The computer addressing system is of the type known as one address. The magnetic drum contains 1,025 registers for computer word storage. Eight of these registers are called fast access registers. In addition to being fast access registers, some fast access registers have special computer functions. Each register in the computer storage system can hold a computer word of 32 binary bits. A computer word can be used to hold either data (numeric or alpha-numeric) or instructions. If a word is used for holding instructions, it will hold two computer instructions per word because each instruction is 16 binary bits. If a computer word is used to hold numeric data, the word can hold 30 binary bits of information (equivalent to nine decimal digits), a bit for use as an overflow test position, and a bit to indicate whether the data is positive or negative. Negative numbers are represented in the computer in two's complement. If a word is used for alpha-numeric storage, it can contain either five characters (each character equal to six binary bits) or six characters (each character equal to five binary bits). Figure 1 gives a representation of the computer words.

Input

Input to the computer can be in any code. From one to three devices in any combination may be used in any Monrobot XI program. The input devices are typewriters, punched paper tape readers, punched card readers, teletype machines, and sixteen-key numeric keyboards.

Output

Output from the computer can be in any code. From one to three devices in any combination may be used in a Monrobot XI program. The output devices are typewriters, paper tape punches, paper card punches, and teletype machines. The output devices may be operated by the program either independently or simultaneously in any combination.

MONROBOT MARK XI COMPUTER WORDS

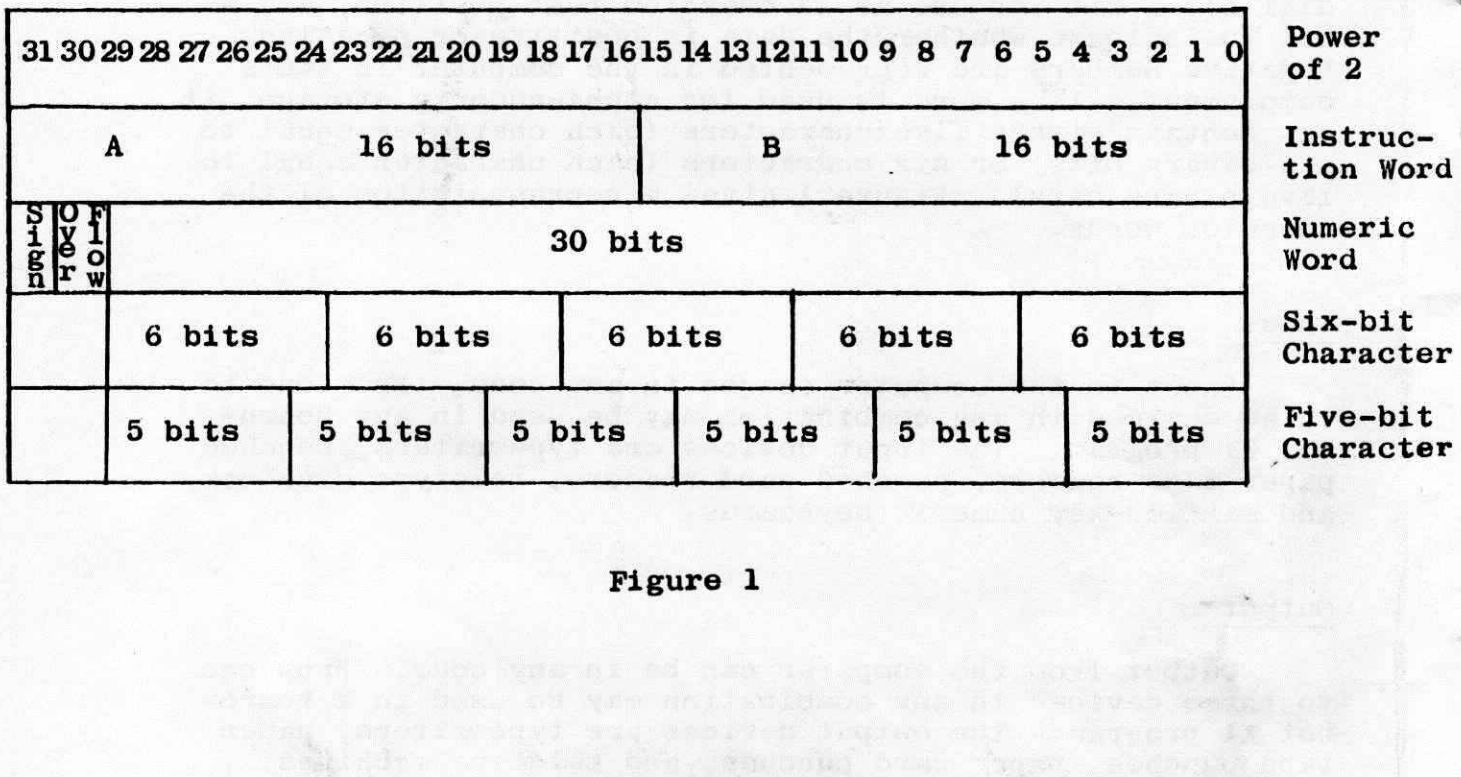


Figure 1

SEXADECIMAL SYSTEM

Although the computer is completely binary and all data within the computer is binary, representation of computer words external to the computer is in the sexadecimal or base 16 system. There are two reasons for this representation. The first reason is that it is easier to recognize and write computer words in sexadecimal than in binary. The other reason is that nonautomatic input to the computer uses the sexadecimal system.

The sexadecimal system has 16 as its number base, just as the decimal system has 10, the octal system eight, and the binary system two. Table 1 gives the decimal, binary, octal, and sexadecimal equivalences.

Decimal	Binary	Octal	Sexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	S
11	1011	13	T
12	1100	14	U
13	1101	15	V
14	1110	16	W
15	1111	17	X

Table 1

Since there are 16 different characters in the sexadecimal system, the six characters over the decimal 10 have to be assigned names and symbols. The naming of the characters is arbitrary. In Monrobot Mark XI these characters were assigned the symbols and names of the English alphabetic letters S through X respectively.

As can be seen from Figure 2, each sexadecimal character represents four binary bits of information. This representation is called a tetrad of information. As each word in the computer holds 32 binary bits, then each word is comprised of eight tetrads.

Tetrad Numbering

In referring to positions within a computer word, a combination of tetrad numbering and binary bit weight is used. Tetrads are numbered from 0 to 7; bits are numbered by their binary weight within the tetrad which can be 8, 4, 2, or 1. Figure 2 gives the tetrad and binary numbering for the computer word.

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	Powers of 2
Tetrad 7	Tetrad 6	Tetrad 5	Tetrad 4	Tetrad 3	Tetrad 2	Tetrad 1	Tetrad 0	Tetrad Number
8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	Binary Weight

Figure 2

In the manual, if a bit position or positions are referred to, they will be denoted by the tetrad number and weight. For example, the high order bit (2^{31}) is referred to as T7-8.

Conversion

This method of numbering the tetrads assigns to each tetrad position its positional exponent, to the base 16, in a sexadecimal number. Table 2 gives the equivalent values for each tetrad position.

Tetrad	Base 16	Base 2	Decimal Equivalent
0	16^0	2^0	1
1	16^1	2^4	16
2	16^2	2^8	256
3	16^3	2^{12}	4096
4	16^4	2^{16}	65536
5	16^5	2^{20}	1048576
6	16^6	2^{24}	16777216
7	16^7	2^{28}	268435456

Table 2

Conversion from decimal to binary and binary to decimal external to the computer becomes a matter of finding the powers of 16 in a number and multiplying those powers by the binary weights of the power. Two examples below show how easily this is accomplished.

Example 1:

Convert the binary number

0000 0000 0000 0000 0010 1001 1101 1010

to decimal. This number is written in sexadecimal as 000029VS. Conversion involves selecting from the table the decimal equivalents of the powers, multiplying by the binary weight of that power, and summing the individual results.

Tetrad	Number	Decimal Equivalent	Binary Weight	Total
7	0	0	0	
6	0	0	0	
5	0	0	0	
4	0	0	0	
3	2	4096	2	8192
2	9	256	9	2304
1	V	16	13	208
0	S	1	10	10
				10714

The result is 10714, the decimal equivalent of the number.

Example 2:

Convert the decimal number 10714 to binary. For conversion from decimal to binary, it is necessary to determine the binary weights of the powers of 16 that are in the decimal number. This is done by dividing the powers of 16 into the number. The quotient of each division gives the binary weights for that power's tetrad position. The remainder gives the value that is used by the next lowest power to obtain its binary weight. This process continues until the zero power or tetrad position is evaluated.

Power	Remainder	Quotient	Tetrad
268435456	10714	0	7
16777216	10714	0	6
1048576	10714	0	5
65536	10714	0	4
4096	10714	2	3
256	2522	9	2
16	218	13	1
1	10	10	0

The result of the conversion is 000029VS when the sexadecimal equivalents are substituted for 13 and 10. Converting the sexadecimal number to binary is now a matter of substituting from Table 1.

These methods give reasonably easy and fast conversion between the number systems. If a Monroe automatic desk calculator is used, this process becomes extremely simple.

Instructions

Instructions in the Monrobot Mark XI are 16 bits or four tetrads long. Therefore, one 32 bit register can hold two instructions. The 16 bits in an instruction word are divided into a command part of six bits and an address part of 10 bits. In the case of instructions that do not have an address, the entire 16 bits are referred to as a command. Figure 3 shows these two cases.

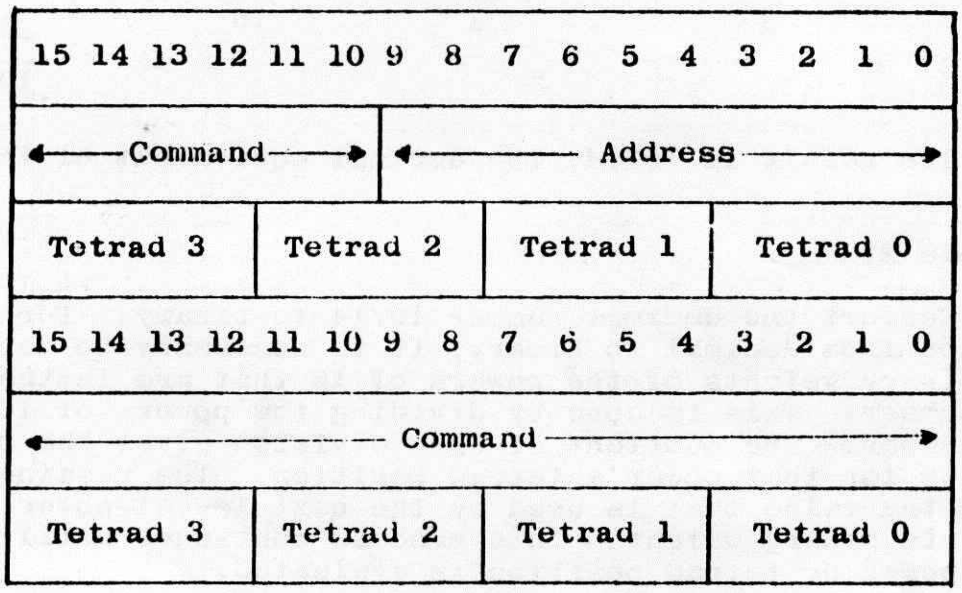


Figure 3

Instructions are written by the programmer not as binary bits but as tetrads. Writing in this manner does not cause any trouble except where the break between command bits and address bits occurs in tetrad 2. In this case sexadecimal values to three refer to the address portion; values above three have also a command portion.

The two instructions per register are referred to as the A step and the B step respectively. The machine always executes the A step first and then the B step. It is never possible to branch to the B step. Figure 4 shows this instruction format.

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Command		Address		Command		Address	
Tetrad 7	Tetrad 6	Tetrad 5	Tetrad 4	Tetrad 3	Tetrad 2	Tetrad 1	Tetrad 0
Step A				Step B			

Figure 4

Table 3 gives the coding in sexadecimal and binary for the instructions that Monrobot Mark XI has in its repertory. The binary coding is for background information to the programmer only. When these instructions are explained in detail later in the manual, only the sexadecimal coding will be given. The commands are divided into two categories consisting of those commands which require storage addresses and those commands which refer to nonstorage operations, such as, input-output and shifting commands.

MONROBOT MARK XI COMMANDS

	Binary																Sexadecimal				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	3	2	1	0	Tetr
Detract	0	0	0	1	0	0	← Address →										1	A	D	R	
Multiply	0	1	0	1	0	0											5	A	D	R	
Store	1	0	1	1	0	0											T	A	D	R	
Interchange	1	1	0	0	0	0											U	A	D	R	≠
Load	1	1	0	1	0	0											V	A	D	R	
Subtract	1	1	1	0	0	0											W	A	D	R	
Add	1	1	1	1	0	0											X	A	D	R	
Extract	1	1	1	1	0	1	← Address →										X	A	D	R	*
Jump	0	0	1	1	0	0	← Address →										3	A	D	R	
Jump Mark	0	0	1	1	0	1											3	A	D	R	*
Jump Zero	0	1	1	0	0	0											6	A	D	R	
Jump High 1	0	1	1	1	0	0	← Address →										7	A	D	R	
Input	0	0	1	0	Address			0	0	0	0	0	0	0	0	0	2	A	0	0	
Output	1	0	1	0	Address			0	0	1	1	1	1	1	1	1	S	B	7	X	
Instruction Output	1	0	1	0	Address			1	← Character →								S	B+1	C	C	
Multiply by 10	1	0	0	0	0	0	0	0	Scale Factor								8	0	P	N	
Divide by 10	1	0	0	0	1	0	0	0									8	8	P	N	
Binary Shift Left	1	0	0	1	0	0	0	0									9	0	P	N	
Binary Shift Right	1	0	0	1	1	0	0	0									9	8	P	N	
Binary End Around	1	0	0	0	1	1	0	0									8	U	P	N	
Binary Shift Right Neg.	1	0	0	1	1	1	0	0	Scale Factor								9	U	P	N	
Intervention Interrogate	1	1	0	0	0	1	0	0	← Address →								U	4	A	D	
Clear FA 6	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	U	5	0	0	
Clear FA 5	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	S	0	0	0	
Set FA 6	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	U	4	0	0	
Stop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

*These instructions add 400 to address to obtain sexadecimal coding.

≠ADR = 0 0 0 $\longrightarrow$ 0 0 5

ADR	=	Address = 0 0 0 $\longrightarrow$ 3 X X
A	=	Input Address = 2, 4, 8
B	=	Output Address = 0, 2, 4, 6, 8, S, U, W
B + 1	=	Instruction Output = 3, 5, 7, 9, T, V, X
CC	=	Any Eight-bit Character
PN	=	Shift from 1 $\longrightarrow$ 8
AD	=	Intervention Interrogate Switch 1 $\longrightarrow$ 8

Table 3

NUMBER SYSTEM

The Monrobot Mark XI is a fixed point binary integer computer. The binary point is located to the right of the number (2^0 position). All numbers used in the computer are treated as integers. Numbers less than one must be scaled so that they are greater than one. Special shift commands are provided to aid in scaling. These shift commands also assist in the decimal-to-binary and binary-to-decimal conversions which must be programmed. Appendix 4 gives methods for conversion using the Monrobot XI shift commands.

The maximum size of a number in the computer is 32 bits, which is equivalent to the decimal number $2^{32} - 1$ (4294967295). The practical size of a number is 30 bits, which is equivalent to the decimal number $2^{30} - 1$ (1073741823). The latter is the number size which will be used when arithmetic operations are discussed in this manual. Negative numbers within the computer are maintained in two's complement form. The high order bit (T7-8) of every numeric register is treated as the sign position. If this bit is zero, the number is regarded as positive. If this bit is one, the number is regarded as negative. In the arithmetic operations of addition, subtraction, and multiplication, the results will be signed correctly if the operands do not exceed 30 bits of information.

Overflow

Numbers which exceed 32 bits as a result of an operation are said to overflow. The computer gives no automatic indication of this overflow. The programmer must test for overflow through the program and prevent numbers from becoming large enough to exceed capacity. When 30 bits are used as the number size, the T7-4 bit position (next to high order) is used as the overflow test position. This position must always have the same value after an arithmetic operation as the sign position (T7-8) or the result exceeds 30 bits in absolute value. In effect, if the result is positive, T7-4 must be zero; if the result is negative, T7-4 must be one. Testing whether overflow has occurred is a function of the programmer through the program. Overflow exceeding 30 bits cannot be permitted because subsequent overflows will change the value of the sign bit from zero to one and one to zero and exceed the capacity of 32 bits. Figure 1 gives a representation of the number word in Monrobot XI.

STORAGE SYSTEM

The Monrobot Mark XI storage system is a magnetic drum. The drum rotates 5,124 times per minute making a drum revolution equivalent to 11.7 milliseconds. The magnetic drum has located on it 1,025, 32-bit registers. Of these registers, 1,017 are in what is called general storage and are available once each drum revolution. The other eight registers are in what is called fast access storage and are available 16 times per drum revolution.

General Storage

The 1,017 general storage registers are divided into 16 tracks along the axis of the drum. Each track is divided into 16 parts called sectors; within each sector are four registers called phases. Figure 5 shows the general storage drum system.

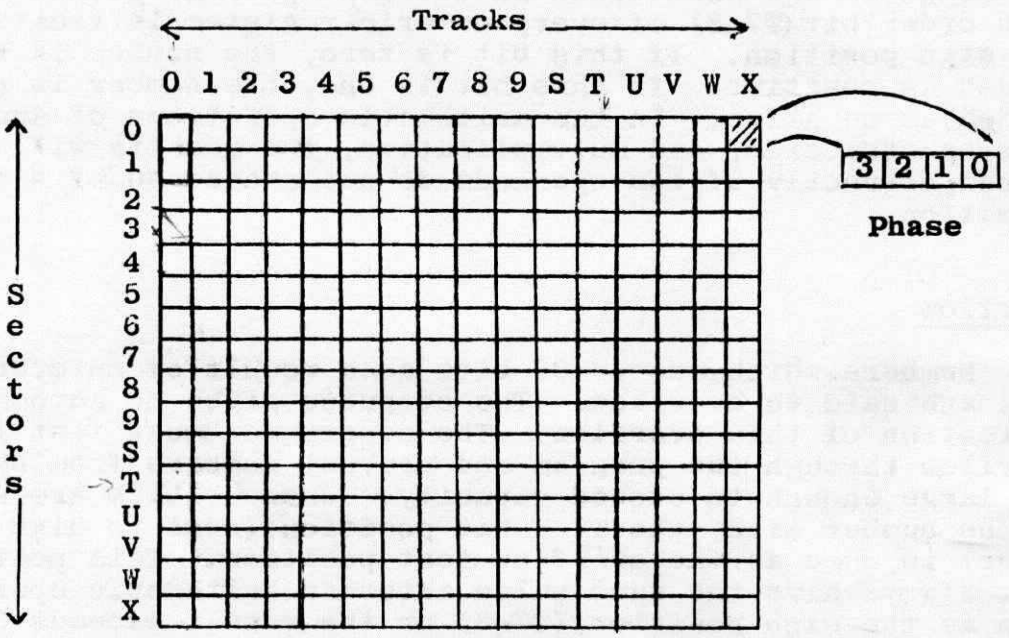

Tracks
0 1 2 3 4 5 6 7 8 9 S T U V W X
Sectors
0 1 2 3 4 5 6 7 8 9 S T U V W X
3 2 1 0
Phase

Figure 5

Addressing a general storage register requires 10 bits. Four bits denote the track, four bits indicate the sector on the track, and two bits indicate the phase within the sector. Figure 6 shows how the address bits are divided within an instruction word.

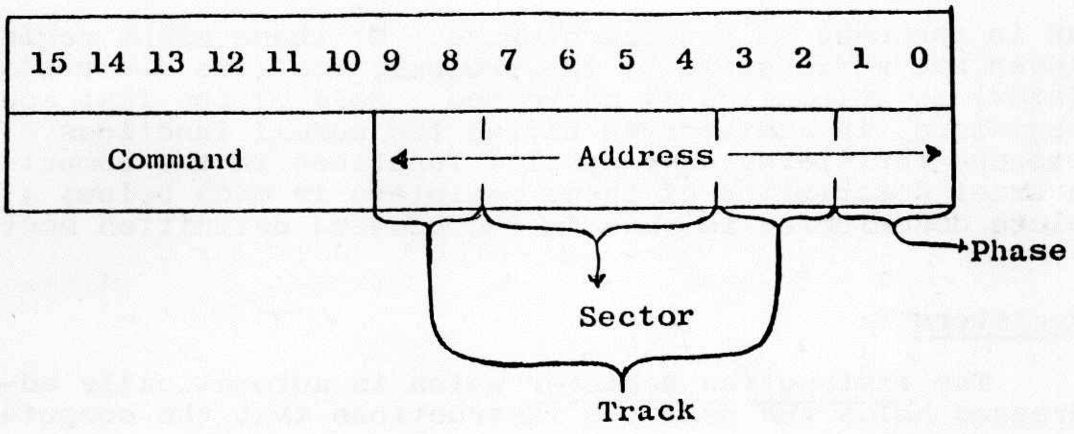


Figure 6

Since all registers are addressed in sexadecimal coding, the programmer does not concern himself with the binary coding of the address. For general information, an example of sexadecimal address coding and its breakdown into the binary address coding is given below.

General storage register 767 in decimal is 2XX in sexadecimal; 2XX = 1011|1111|11 in binary. This register is located in track T (1011) sector X (1111) and phase 3 (11).

Access Time

Every general storage track has a device called a read-write head for obtaining data from a register and recording data into a register. Each sector around the track is presented to this read-write head once every drum revolution. When an instruction calls for a general storage address, the computer cannot read from the register or write into the register until the sector in which that register is located is under the read-write head. The time it takes from when the instruction is given to the time when the sector is available is called access time. Access time can range from zero sector times (sector is under read-write head) to 15 sector times (sector is a complete drum revolution away). Average access time to general storage registers is eight word times.

Fast Access Storage

The magnetic drum has two tracks which hold four registers each. These registers are called fast access registers because they are available every sector time (16 times a drum revolution) rather than once a drum revolution as in the case of general storage. Of these eight registers, seven are addressable by the program, one (the instruction register) is automatically addressed. Some of the fast access registers, in addition to having the normal functions of storage registers, have special functions in the computer. A brief description of these registers is made below; a complete description is made in the command definition sections.

Registers

The Instruction Register which is automatically addressed holds the next two instructions that the computer may execute.

Fast Access 6 (address 0 0 6) performs the function of the accumulator. It holds one of the operands and most of the results in arithmetic operations. It receives the input character in input operations, holds the high order portion of the product in multiplication, and the remainder in detraction.

Fast Access 5 (address 0 0 5) holds the low order portion of the product in multiplication, the number of detractions in the detract operation, the output character in the output operation, and is part of the end around shift.

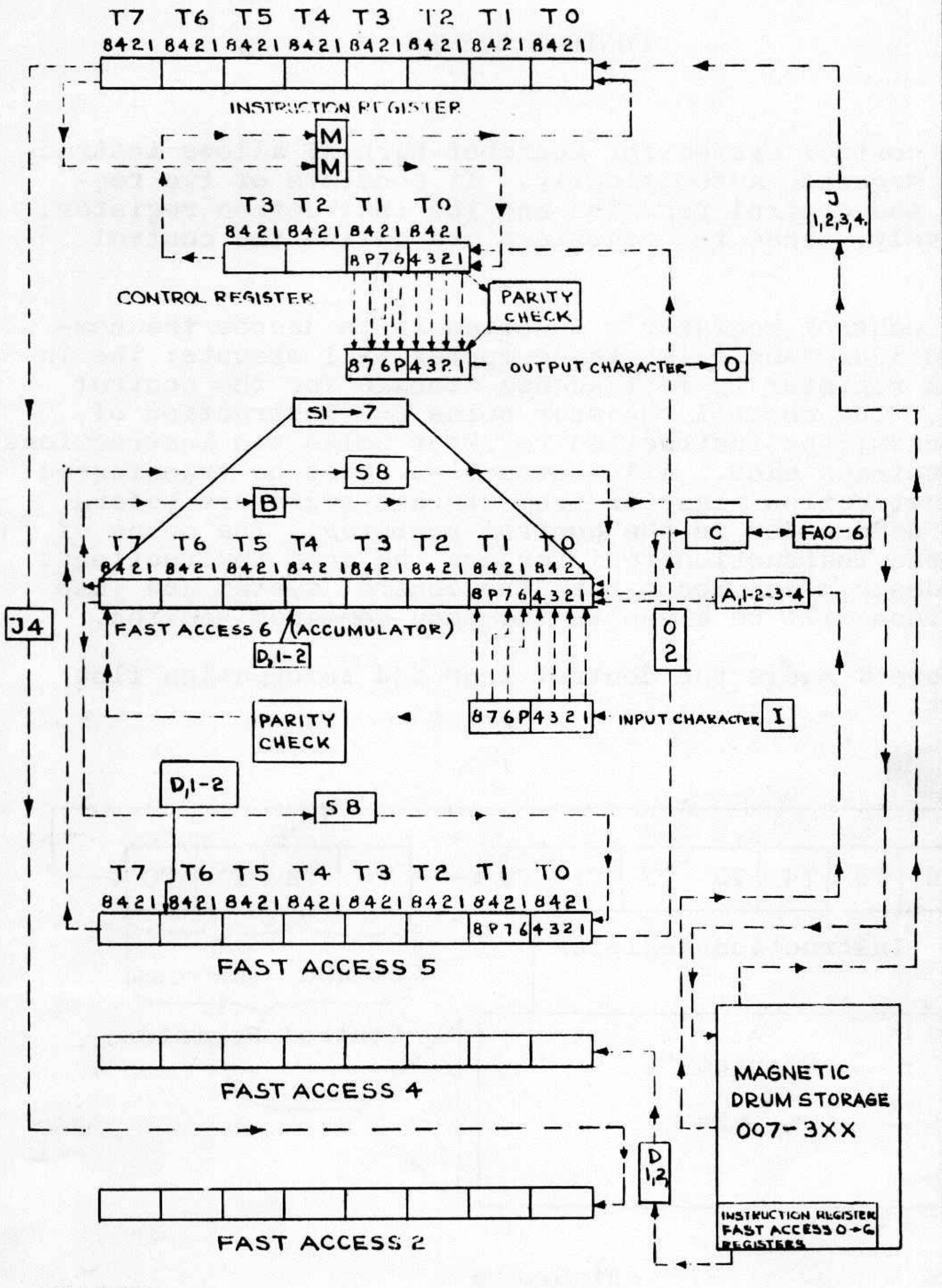
Fast Access 4 (address 0 0 4) holds the multiplicand in the multiply operation and the detractor in the detract operation.

Fast Access 2 (address 0 0 2) receives the contents of the instruction register in the jump mark instruction.

Fast Access Registers 3 (address 0 0 3), 1 (address 0 0 1), 0 (address 0 0 0) have no special functions and can always be used as storage registers. Figure 7, which is the system diagram of Monrobot Mark XI, shows the flow of information between these registers and the rest of the computer.

Computer Timing

All timing in the computer is based on the drum revolution speed and the number of sectors in a revolution. Since there are 16 sectors and a drum speed of 11.7 milliseconds, a sector time is equivalent to .73 milliseconds. All computer operations are measured in sector times. The minimum instruction time is four sector times. The actual length of time depends upon two factors: execution time and access time. Execution time is the time necessary to execute the command; access time is the time necessary to obtain the contents of the addressed register. Appendix I gives data on execution times for each command and page 72 gives rules for minimizing access time.



COMMANDS

A	B	C	J	S	O
1.- LOAD	1.- STORE	INTERCHANGE	1.- JUMP	1.- DECIMAL SHIFT RIGHT	1.- INSTRUCTION OUTPUT
2.- SUBTRACT	D	FA6↔FAn	2.- JUMP ZERO	2.- DECIMAL SHIFT LEFT	
3.- ADD	1- DETRACT	n = 0→6	3.- JUMP T7,8=1	3.- BINARY SHIFT RIGHT	2.- FA5 OUTPUT
4.- EXTRACT	2- MULTIPLY		4.- JUMP MARK	4.- BINARY SHIFT LEFT	
				5.- DIVIDE BY 2^n	
				6.- CLEAR FA6	
				7.- INTERROGATE SWITCH	
				8.- END AROUND SHIFT	

I	M
1- INPUT	1.- INSTRUCTION SEQUENCE

MONROBOT MARK XI SYSTEM DIAGRAM

FIGURE 7

CONTROL SYSTEM

The control system for Monrobot Mark XI allows instructions to sequence automatically. It consists of two registers: the control register and the instruction register. Collectively, these two registers are called the control loop.

The control register's function is to decode the commands and addresses which the computer will execute; the instruction register is fast access storage for the control register. The control register holds one instruction of four tetrads; the instruction register holds two instructions of four tetrads each. All instructions must be transferred to the instruction register from storage registers before they can be decoded in the control register. The means of loading the instruction register are the jump instructions. Further description about both the control system and jump instructions will be given in the jump commands section.

Figure 8 shows the control loop and information flow within it.

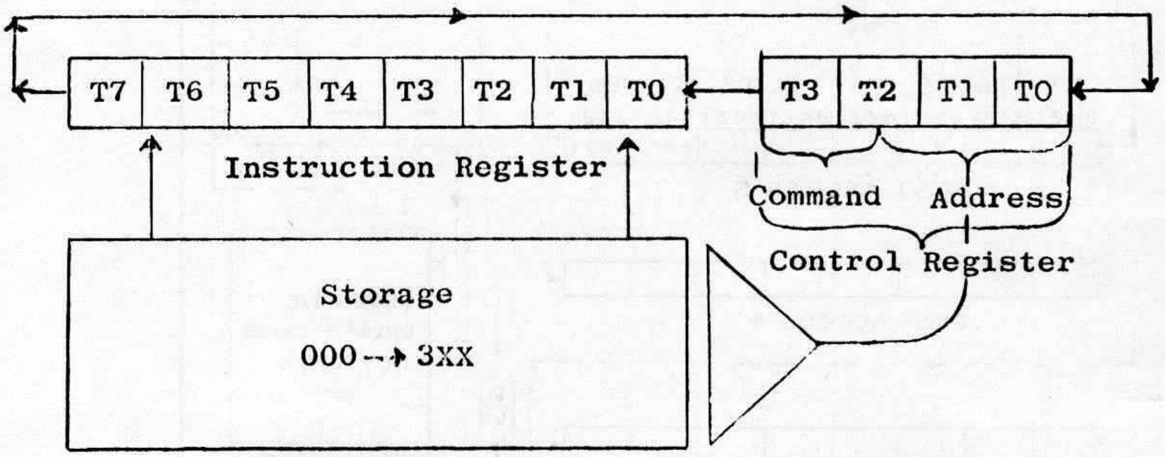


Figure 8

MONROBOT XI COMMAND DESCRIPTION

General

For this manual Monrobot XI commands have been divided into two groups, namely, commands that require a storage address and commands that do not require a storage address. Each of these two groups has again been subdivided. Addressable commands are either arithmetic or control commands. Nonaddressable commands are either shifting commands or input-output commands. Arithmetic commands will be described first, then control, input-output, and shifting commands.

Format

Each command will be described in sexadecimal notation. The following format will be used.

Command	English words for command.
Code	Sexadecimal coding of command.
Instruction	Sexadecimal code of command and variations to command. Variations are mnemonic with range of variations written on the right.
Fast Access Register	This will give the fast access registers which the command uses or may affect by changing the values of these registers after the operation.
Description	English language description of command. The term FA used in the description refers to fast access.
Example	This gives a coded example of the command with values in the affected registers both prior to the operation and subsequent to the operation. These values are in decimal and sexadecimal notation.

Command	Add	
Code	X - - -	
Instruction	X A D R	A D R = 0 0 0 ⟶ 3 X X

Fast Access Register

0 0 6

Description

The add command adds the contents of the addressed storage register to the contents of FA 6 (accumulator) and places the sum in FA 6 (accumulator). The addressed storage register is unchanged by this command. Negative and positive numbers can be added, and the correct algebraic result will occur. Carries exceeding 30 bits will affect the overflow test position (T7-4), carries exceeding the overflow test position will affect the sign position (T7-8), and carries beyond this position are lost.

Example

	Register	Decimal	Sexadecimal
Instruction	X 0 0 3		
Before Add	0 0 3	4096	1000
	0 0 6	256	100
After Add	0 0 3	4096	1000
	0 0 6	4352	1100
Instruction	X 0 0 6		
Before Add	0 0 6	999999999	3T9SU9XX
After Add	0 0 6	1999999998	773593XW

This example shows how overflow affects the overflow test position.

Command	Subtract	
Code	W - - -	
Instruction	W A D R	A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Register

0 0 6

Description

The subtract command subtracts the contents of the addressed storage register from the contents of FA 6 (accumulator) and places the remainder in FA 6 (accumulator). The addressed register is unchanged by this command.

Both negative and positive numbers can be subtracted, and the correct algebraic result will occur. Negative remainders will be expressed in two's complement. Borrows exceeding 30 bits will affect the overflow test position (T7-4), borrows exceeding the overflow test position will affect the sign position (T7-8), and borrows beyond the sign position are lost.

Example

	Register	Decimal	Sexadecimal
Instruction	W 1 3 U		
Before Subtract	1 3 U	25	19
	0 0 6	52	34
After Subtract	1 3 U	25	19
	0 0 6	27	1T
Instruction	W 2 X X		
Before Subtract	2 X X	346	15S
	0 0 6	345	159
After Subtract	2 X X	346	15S
	0 0 6	- 1	XXXXXXXX

Command	Detract
Code	1 - - -
Instruction	1 A D R A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Registers

004, 005, 006

14

Description

The detract command transfers the contents of the addressed register automatically to FA 4. It then subtracts the contents of FA 4 from the contents of FA 6 (accumulator) until FA 6 is less than the contents of FA 4. For every such subtraction that is made, a one is added to the low order (TO-1) position of FA 5. The contents of both FA 6 and FA 4 are considered as positive numbers. If the contents of the addressed register are zero when this command is used, the detract operation will not cease unless the computer is reset. The contents of the addressed storage register are unchanged by this command.

Examples

Instruction 1 0 6 3

	Register	Decimal	Sexadecimal
Before Detract	0 6 3	5	5
	0 0 6	321	141
	0 0 5	0	0
	0 0 4	652	28U
After Detract	0 6 3	5	5
	0 0 6	1	1
	0 0 5	64	40
	0 0 4	5	5

Command	Multiply	
Code	5 - - -	
Instruction	5 A D R	A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Registers

004, 005, 006

Description

The multiply command transfers the contents of the addressed register automatically to FA 4. It then multiplies the contents of FA 4 by the contents of FA 6 (accumulator) to develop a 64-bit product. The low order 32 bits of this product are in FA 5, and the high order 32 bits are in FA 6. If the product is negative, it will be represented in two's complement in both FA 5 and FA 6. The contents of the addressed register are not affected by this command.

Examples

Instruction	5 2 3 X		
	Register	Decimal	Sexadecimal
Before Multiply	2 3 X	350	15W
	0 0 6	15	X
	0 0 5	6721	1S41
	0 0 4	621	26V
After Multiply	2 3 X	350	15W
	0 0 6	0	0
	0 0 5	5250	1482
	0 0 4	350	15W

Instruction	5 0 0 6		
Before Multiply	0 0 6	65536	10000
	0 0 5	216	V8
	0 0 4	1248	4W0

	Register	Decimal	Sexadecimal
After Multiply	0 0 6	1	1
	0 0 5	0	0
	0 0 4	65536	10000
Instruction	5 3 W W		
Before Multiply	3 W W	999999999	3T9SU9XX
	0 0 6	999999999	3T9SU9XX
	0 0 5	0	0
	0 0 4	10	S
After Multiply	3 W W	999999999	3T9SU9XX
	0 0 6	232830643	VWOT6T3
	0 0 5	808348673	302W6U01
	0 0 4	999999999	3T9SU9XX

Command	Store	
Code	T - - -	
Instruction	T A D R	A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Registers

0 0 6

Description

The store command replaces the contents of the addressed storage register with the contents of FA 6 (accumulator). FA 6 is unchanged by the store command. The store command is the only command that will change the contents of register 0 0 7 $\longrightarrow$ 3 X X.

If the address of the store instruction is 0 0 6, the contents of FA 6 will replace the contents of FA 4, but the contents of FA 6 will remain unchanged.

Example

Instruction T 3 2 3

	Register	Decimal	Sexadecimal
Before Store	3 2 3	625	271
	0 0 6	15630	3VOW
After Store	3 2 3	15630	3VOW
	0 0 6	15630	3VOW

Command	Load
Code	V - - -
Instruction	V A D R A D R = 0 0 0 → 3 X X

Fast Access Register

0 0 6

Description

The load command replaces the contents of FA 6 (accumulator) with the contents of the addressed storage register. The addressed storage register is unchanged by this command.

Example

Instruction V 0 6 0

	Register	Decimal	Sexadecimal
Before Load	0 6 0	345189	54465
	0 0 6	0	0
After Load	0 6 0	345189	54465
	0 0 6	345189	54465

Command Interchange

Code U - - -

Instruction U A D R $\qquad$ A D R = 0 0 0 $\longrightarrow$ 0 0 6

Fast Access Registers

0 0 6 and addressed FA register

Description

The interchange command replaces the contents of the addressed fast access register with the contents of FA 6 (accumulator) and places the contents of the addressed fast access register into FA 6 (accumulator).

If the address of this command is 0 0 6, the contents of FA 6 will replace the contents of FA 4. However, the contents of FA 6 will remain unchanged.

Example

Instruction U 0 0 1

	Register	Decimal	Sexadecimal
Before Interchange	0 0 6	25002	61SS
	0 0 1	20	14
After Interchange	0 0 6	20	14
	0 0 1	25002	61SS

Command	Extract
Code	X 4 - -
Instruction	X (4+A) D R A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Register

0 0 6

Description

The extract command compares the 32 bit positions in the addressed storage register with the 32 bit positions in FA 6 (accumulator). Wherever a one occurs in corresponding bit positions, a one is placed in that bit position in FA 6; wherever a one is not present in both positions, a zero is placed in that bit position. The addressed storage register is not changed by this command.

This command is a logical multiplication command.

Example

Instruction X 5 6 3

	Register	Decimal	Sexadecimal
Before Extract	1 6 3	Not	11V7
	0 0 6	Applicable	X9
After Extract	1 6 3		11V7
	0 0 6		V1

Instruction	X 4 0 2	
Before Extract	0 0 2	XXXX0000
	0 0 6	20093200
After Extract	0 0 2	XXXX0000
	0 0 6	20090000

CONTROL INSTRUCTIONS

Monrobot Mark XI has four control or jump instructions which permit conditional and unconditional branching of program control. Prior to describing the four jump instructions, a further description of the control loop will be made to facilitate an understanding of instruction sequencing in the computer. Figure 9 shows the control loop which consists of the control register and instruction register.

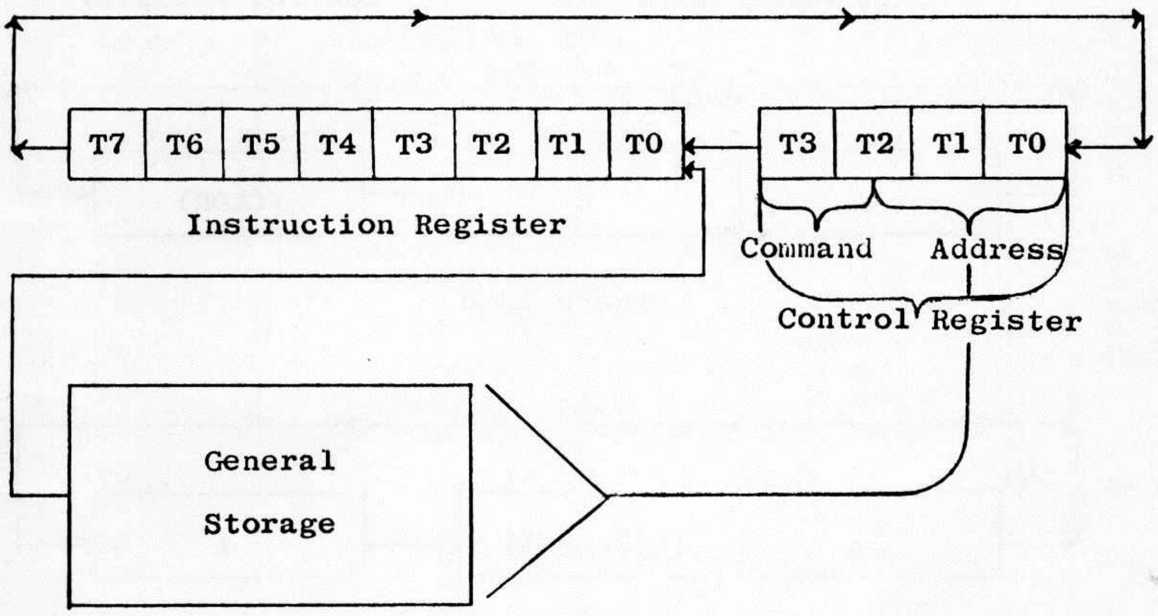


Figure 9, Control Loop

The function of the control register is to decode instructions; the function of the instruction register is to provide fast access storage for the control register. The instruction register holds the instructions that will be executed or have been executed. All instructions must be transferred to the instruction register from storage before they can be executed by the control register. The jump commands are the commands that do this transfer.

As can be seen from Figure 9, three instructions are present in the control loop. Two of these instructions are in the instruction register; one (currently being executed) is in the control register. Of these three instructions one must always be a jump instruction in order to load the instruction register with the next two instructions in the program sequence.

Program Sequencing

Program sequencing operates by the jump instruction bringing the contents of the desired register into the instruction register, executing the two instructions in the control register, and then automatically bringing the next register in sequence into the instruction register unless one of the instructions executed was a jump instruction. This jump instruction would have loaded the contents of the instruction register with two instructions and started a new automatic sequence.

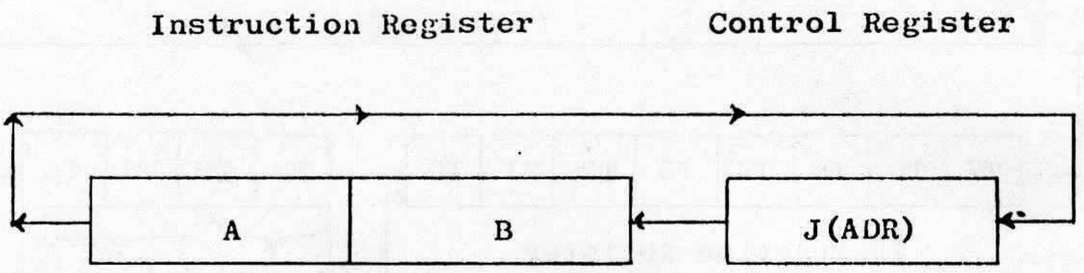


Figure 10.0

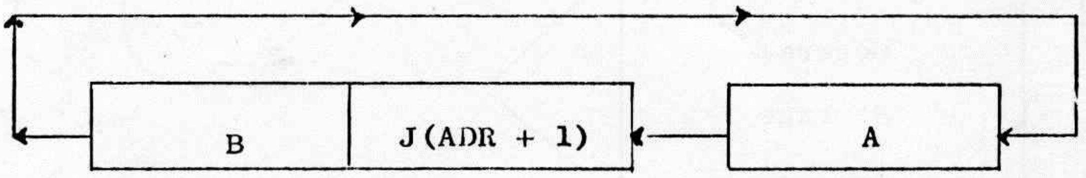


Figure 10.1

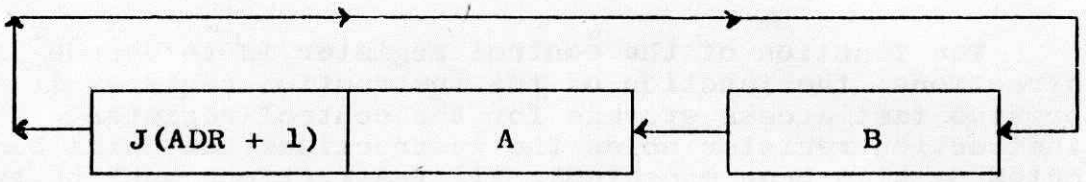


Figure 10.2

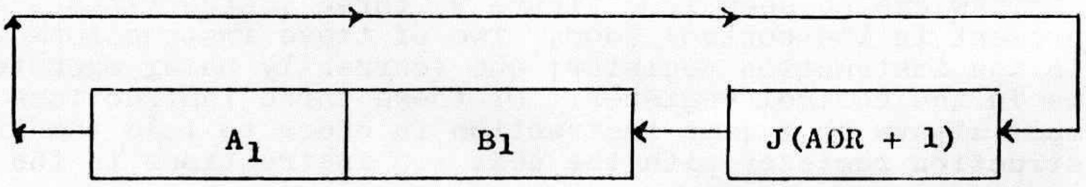


Figure 10.3

The Figures 10.0 through 10.3 show how Monrobot Mark XI program sequencing operates. Figure 10.0 shows a jump instruction in the control register. This jump instruction has loaded the instruction register with the contents of the storage register addressed by the jump command. The computer then automatically shifts the A step from the instruction register into the control register. The B step moves up to replace the A step. The jump instruction has a one automatically added to its address portion and is moved into the positions vacated by the B step. Figure 10.1 shows this operation. After the A step has been executed, the B step is shifted into the control register. The augmented jump command moves into the vacated B step positions, and the executed A step into the jump position. Figure 10.2 shows this operation. Once the B step has been executed, the entire process is repeated with the control register being loaded with the jump command with its address increased by one. Figure 10.3 gives this operation. In this manner automatic sequencing will continue unless either the A step or the B step contains a jump instruction. This jump command will load the instruction register as in Figure 10.0 and then have its address augmented automatically to start a new automatic sequence.

Although three instructions are constantly circulating in the control loop, the only time a jump instruction is written by the programmer is when the control sequence is to be changed from being automatically sequenced by the computer.

On the execution of instructions from the instruction register, the A step is always done first and then the B step. It is never possible to alter this sequence. Figures 10.0 to 10.3 show this sequencing.

The jump command which does the automatic sequencing is the unconditional jump. Any other jump commands which may load the instruction register become unconditional jumps in the control loop after they have loaded the instruction register.

Command	Jump Unconditional	
Code	3 - - -	
Instruction	3 A D R	A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Registers

Instruction

Description

The jump unconditional command replaces the contents of the instruction register with the contents of the addressed register. The previous sequence of instructions is interrupted, and a new sequence starting with the A step of the addressed register is started. The contents of the addressed register are unchanged by this command.

Example

Instruction 3 2 0 0

	Register	Sexadecimal
Before Jump	2 0 0	V002T315
	Instruction	X001W216
After Jump	2 0 0	V002T315
	Instruction	V002T315

Command	Jump Mark	
Code	3 4 - -	
Instruction	3 (4+A) D R	A D R = 0 0 0 $\longrightarrow$ 3 X X

Fast Access Registers

Instruction, 0 0 2

Description

The jump mark command replaces the contents of FA 2 with the contents of the instruction register and then replaces the contents of the instruction register with the contents of the addressed register. The previous sequence of instructions is interrupted and a new sequence beginning with the A step of the addressed register is started. The contents of the addressed register are unchanged by this command.

Examples

Instruction	3 6 1 2	
	Register	Sexadecimal
Before Jump Mark	2 1 2	X00530XX
	Instruction	320XX001
	0 0 2	00000036
After Jump Mark	2 1 2	X00530XX
	Instruction	X00530XX
	0 0 2	320XX001
Instruction	3 5 X X	
Before Jump Mark	1 X X	V301T001
	Instruction	W09X3213
	0 0 2	00000000
After Jump Mark	1 X X	V301T001
	Instruction	V301T001
	0 0 2	W09X3213

The jump mark command permits program branching with memory of the program register which caused the program branch. FA 2 contains this memory. The memory consists of the contents of the instruction register when the jump mark command was being executed in the control register. One of the instructions in FA 2 after the jump mark command is the executed A step if the jump mark was the B step or the unexecuted B step if the jump mark command was the A step of the program register which contained the jump mark command. The other instruction is the automatic jump instruction which is always present in the control loop. The address of this automatic jump instruction is one more than the program register which contained the jump mark. This automatic jump instruction would have continued program sequencing if the jump mark instruction had not been written as either the A or B step. Since this jump is transferred to FA 2 by the jump mark command, any subsequent jump to FA 2 will retransfer program control to the program register after the program register which contained the jump mark command. If the jump mark command was the A step of the program register, the unexecuted B step would be the first instruction performed when control was returned to FA 2. The automatic jump which actually returns the program to the desired sequence would be executed after this step. If the jump mark was in the B step only the automatic jump is executed when control is returned to FA 2.

It is not necessary to keep the marked instruction in FA 2. The marked instruction can be transferred to any storage register, allowing FA 2 to be used for working storage or receiving more mark instructions. To return to the previous sequence, the program would jump to the register where the contents of FA 2 had been transferred.

The jump mark instruction makes possible effective use of subroutines in Monrobot Mark XI. Entrance to the subroutines is made through the jump mark instruction. Exit from the subroutines is a jump to the register where the mark has been placed. The program then resumes its sequence at the instruction step after the jump mark instruction.

Command	Jump on Zero	
Code	6 - - -	
Instruction	6 A D R	A D R = 0 0 0 → 3 X X

Fast Access Registers

Instruction

Description

The jump on zero command replaces the contents of the instruction register with the contents of the addressed storage register if the contents of FA 6 (accumulator) are equal to zero. The previous sequence of instructions is interrupted and a new sequence starting with the A step of the addressed register is begun.

If the contents of FA 6 (accumulator) are not equal to zero, the program continues by executing the next instruction from the instruction register.

This command, whether successful or unsuccessful, does not change the contents of the addressed storage register.

Examples

Instruction 6 3 2 5

	Register	Sexadecimal
Before Jump on Zero	3 2 5	T0013096
	Instruction	3300X002
	0 0 6	0
After Jump on Zero	3 2 5	T0013096
	Instruction	T0013096
	0 0 6	0

Instruction 6 1 X X

	Register	Sexadecimal
Before Jump on Zero	1 X X	V2TTX001
	Instruction	3205X005
	0 0 6	12
After Jump on Zero	1 X X	V2TTX001
	Instruction	3205X005
	0 0 6	12

Negative.

Command	Jump on High Order 1
Code	7 - - -
Instruction	7 A D R A D R = 0 0 0 → 3 X X

Fast Access Registers

Instruction

Description

The jump on high order 1 command replaces the contents of the instruction register with the contents of the addressed storage register if and only if the high order bit (T7-8) of FA 6 (accumulator) is equal to one. The previous sequence of instructions is interrupted, and a new sequence is begun starting with the A step of the addressed register. If the high order bit (T7-8) of FA 6 (accumulator) is equal to zero, the program continues by executing the next instruction from the instruction register.

This command, whether successful or unsuccessful, does not change the contents of the addressed storage register.

Example

Instruction	7 0 6 0	
	Register	Sexadecimal
Before Jump High 1	0 6 0	U005X004
	Instruction	3261T050
	0 0 6	80000000
After Jump High 1	0 6 0	U005X004
	Instruction	U005X004
	0 0 6	80000000
Instruction	7 3 0 0	
Before Jump High 1	3 0 0	X012T005
	Instruction	X0013310
	0 0 6	00005602

	Register	Sexadecimal
After Jump High 1	3 0 0	X012T005
	Instruction	X0013310
	0 0 6	00005602

In most cases this command can be thought of as a jump on negative since all negative numbers must have a high order bit. However, shift commands and the input command also affect the high order bit position of FA 6 so that the jump on high 1 is more general.

INPUT-OUTPUT

General

The Monrobot Mark XI input-output unit is the character. Every time an input or output command is given, one character which may be up to two tetrads (eight bits) is either read into the computer from an input device or written on the output devices. The programmer is responsible through the program for the conversion, transfer, and manipulation of all characters so that they form computer words on input, and for the inverse, that computer words form characters on output. Special shift commands (described in section "Shift Commands," page 47) are provided to aid the programmer in performing this program function. Appendix 4 gives some conversion and manipulation methods.

The Monrobot Mark XI system can have from one to three input and output devices in operation in the same program. Only one input device can read a character into the computer during an input command; however, characters may be sent to one, two, or three devices simultaneously during an output command.

All characters read into or out of the computer have parity bits. Parity is defined as odd parity. That is, the sum of the one bits in each character must equal an odd number. All characters read out of the computer have odd parity assigned to them. Input characters that do not have odd parity are singled out to the program.

The position of bits in the input-out character is represented differently when the character is within the Monrobot Mark XI than when the character is external to the device. Characters external to the Monrobot Mark XI have the following form, where Lines A and C give the binary weight of the bit position. Line B gives the numeric order of the bits reading from right to left where P represents the location of the parity bit.

A	8 4 2 1	8 4 2 1	Binary Weight
B	8 7 6 P	4 3 2 1	Numeric Bit Position
C	T1	T0	Tetrad Position

Characters internal to the Monrobot Mark XI have the following form where Lines A and C give the binary weight of

A	8 4 2 1	8 4 2 1
B	8 P 7 6	4 3 2 1
C	T1	T0

the bit position. Line B gives the position of the character bits of the external character when automatically rearranged by the computer for its internal use. The P position is shifted to the T1-4 position and is automatically set to zero. The T1-4 and T1-2 positions of the external character are shifted to the T1-2 and T1-1 positions respectively. The other positions remain exactly the same. Figure 11 shows this rearrangement.

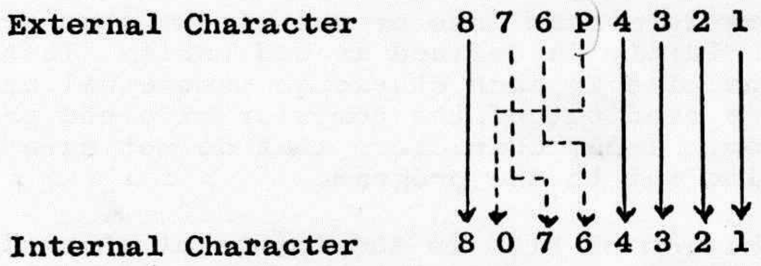


Figure 11

This rearrangement allows most characters to enter the computer as six bit characters and allows for maximum density of packing characters into a computer register.

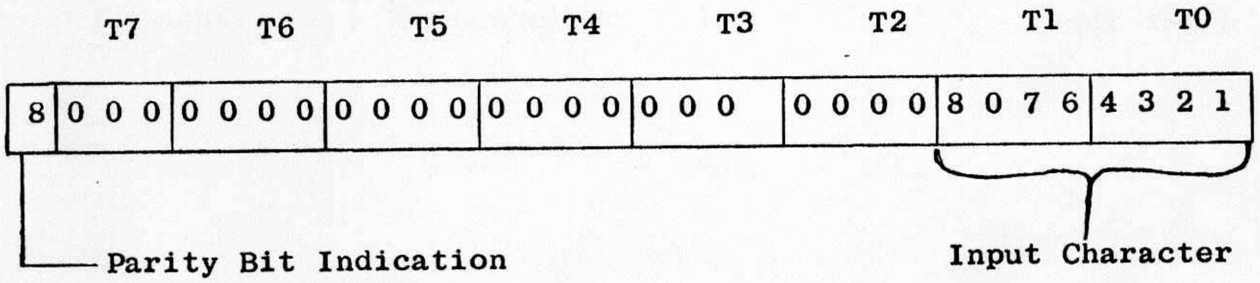
<u>Command</u>	Input		
<u>Code</u>	2 2 0 0	Device 1	
	2 4 0 0	Device 2	
	2 8 0 0	Device 3	
<u>Instruction</u>	2 A 0 0		A = 2, 4, 8.

<u>Fast Access Registers</u>

0 0 6

<u>Description</u>

The input command replaces the contents of FA 6 (accumulator) with the character that is at the input device specified by the instruction. The character which may be up to eight bits (two tetrads) is placed in the low order (T0 and T1) positions of FA 6. All other bit positions are set to zero automatically with the exception of the high order bit position (T7-8). This position is set to one if the input character has even parity; it is set to zero if the input character has odd parity. Figure 12 shows FA 6 for this command.



FA 6 on Input

Figure 12

The character read into the T0-T1 tetrads has its bit positions rearranged as indicated from its input format of 876P4321. When an input command addresses a device, the addressed device must have a character available or the computer will wait at that input device until a character is made available. No other computer operations can proceed until the input command has been executed.

Example

Input from Device 3

Instruction	2 8 0 0		
	Register	Sexadecimal	Binary Character Representation
Before Input	0 0 6	XXX00000	01100001
After Input	0 0 6	00000031	00110001

Input from Device 2, Parity Error

Instruction	2 4 0 0		
Before Input	0 0 6	00000000	00000011
After Input	0 0 6	80000003	00000011

Input from Device 1, No Parity Error

Instruction	2 2 0 0		
Before Input	0 0 6	012W67X3	00010011
After Input	0 0 6	00000003	00000011

Command	FA 5 Output Unkown.	
Code	S 0 7 X	No Device. Clear FA 5
	S 2 7 X	Device 1
	S 4 7 X	Device 2
	S 6 7 X	Device 1 and 2
	S 8 7 X	Device 3
	S S 7 X	Device 1 and 3
	S U 7 X	Device 2 and 3
	S W 7 X	Device 1, 2, and 3.
Instruction		
	S B 7 X	B = 0, 2, 4, 6, 8, S, U, W.

Fast Access Registers

0 0 5

Description

The FA 5 output command takes the low order two tetrads (T0, T1) of fast access register 5 and sends them to the device or devices specified by the T2 tetrad of the instruction. The command then sets the contents of FA 5 to zero. The character read out of the computer has its bits rearranged by the command to 876P4321 from the internal format of 8P764321. The computer automatically computes a parity bit for output if the T1-4 bit of the two tetrads is zero. A parity bit of one will be inserted into the character if the sum of the bits is even; if the sum of the bits is odd, a zero will be inserted. However, if the T1-4 bit is equal to one, a one is always inserted into the parity bit position on output.

If the output device or devices that the output character is sent to are not ready or available to receive the character, the computer will wait at these devices holding the character until the device or devices are ready or available. No other computer operation will proceed until the output command has been executed.

Example

Output to Devices 2 and 3

Instruction	S U 7 X		
	Register	Sexadecimal	Binary Character Representation
Before Output	0 0 5	00000013	00010011
After Output	0 0 5	00000000	00100011

Output to Devices 1, 2, and 3

Instruction	S W 7 X		
Before Output	0 0 5	00000041	01000001
After Output	0 0 5	00000000	00010001

Command	Instruction Output Known.	
Code	S 3 - -	Device 1
	S 5 - -	Device 2 TAPE. output -
	S 7 - -	Device 1 and 2
	S 9 - -	Device 3
	S T - -	Device 1 and 3
	S V - -	Device 2 and 3
	S X - -	Device 1, 2, and 3
Instruction	S P C C	P = 3, 5, 7, 9, T, V, X
		CC = 0 0 $\rightarrow$ X X

Fast Access Registers

None

Description

The instruction output command takes the T0 and T1 tetrads of the instruction and sends them to the output devices specified by the P tetrad where they become output characters. The format of the output character is rearranged automatically on output from 8P764321 within the instruction to 876P4321 when actually sent to the output device. Parity is not computed for this output; however, the programmer can assign parity by making the P position equal to one if the sum of the other character bits is even. The P position should be zero if the sum of the remaining bits is odd.

If the output device or devices that the output character is sent to are not ready or available to receive the character, the computer will wait at these devices holding the character until the device or devices are ready or available. No other computer operations can proceed until the instruction output command is executed.

Example

Output to Device 1

Instruction S 3 6 2

	Register	Binary Character Representation
Before Output	None	01100010
After Output	None	01010010

Output to Device 1 and 2

Instruction S 7 0 1

	Register	Binary Character Representation
Before Output	None	00000001
After Output	None	00000001

SHIFT COMMANDS

General

The Monrobot Mark XI has six types of shift commands. These commands are designed to shift data and to aid in the binary-decimal conversions. These shift commands, which will be described in detail below, are:

(1) Decimal Shift Left (multiply by 10^1 to 10^8).

(2) Decimal Shift Right (divide by 10^1 to 10^8).

(3) Binary Shift Left (multiply by 2^1 to 2^8).

(4) Binary Shift Right (divide by 2^1 to 2^8).

(5) Binary Left End Around Shift (multiply FA 6 and FA 5 by 2^1 to 2^8).

(6) Binary Shift Right Maintain High Order Bits (divide positive or negative by 2^1 to 2^8).

The shift commands always operate on the contents of FA 6 (accumulator) and also in one case (binary left end around shift) FA 5. Consequently, each shift instruction has the following form

8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1
T3	T2	T1	T0
Command		Number of Shifts	

where the T3 and T2 tetrads give the command code for the type of shift and the T1 and T0 tetrads give the power (n^1 to n^8) by which the information is to be shifted.

The sexadecimal coding for each shift command is given in Table 3. The sexadecimal coding for the number of shifts desired is given in Table 4. A shift of zero units is not permissible. If this shift is coded, the computer will cycle endlessly until operator intervention resets the computer.

The sign of numeric data is not preserved by the shift instructions with the exception of the binary shift right maintain high order bits. The sign bit position (T7-8) in FA 6 is shifted by these commands exactly as all the other bit positions.

Shift Command Coding

Command	Code T3 T2
Decimal Shift Left	8 0
Decimal Shift Right	8 8
Binary Shift Left	9 0
Binary Shift Right	9 8
Binary Left End Around	8 U
Binary Right Maintain High Order	9 U

Table 3

Power Shift Coding

Number Power	Code T1 T0
1	0 1
2	0 2
3	0 4
4	0 8
5	1 0
6	2 0
7	4 0
8	8 0

Table 4

In the description of the shift commands, tetrads T1 and T0 are referred to as the PN tetrads.

Command	Decimal Shift Left (multiply by 10^1 to 10^8)
Code	8 0 - -
Instruction	8 0 P N PN = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6

Description

The decimal shift left command multiplies the contents of FA 6 (accumulator) by the power of ten specified by the PN tetrads of the instruction. The powers of ten range from one to eight. The result of the multiplication appears in FA 6. If the result of the multiplication exceeds the 32 bit capacity of FA 6, the overflow is lost off the high order end (T7-8).

While this command is used as a decimal shift left, care must be exercised in its use. It is actually a binary multiplication of FA 6 by powers of ten. When the shifted number is decimal, the shift may produce bits in T7-4 and T7-8 because of the 32 bit register capacity. If the shifted number were evaluated subsequently, it would be greater by the 2^{31} and 2^{30} bits than a number produced by a true decimal shift.

Examples

Shift by 10^1

	Register	Decimal	Sexadecimal
Instruction	8 0 0 1		
Before Decimal Shift Left	0 0 6	10	S
After Decimal Shift Left	0 0 6	100	64

Shift by 10^4

Instruction	8 0 0 8		
Before Decimal Shift Left	0 0 6	25	19
After Decimal Shift Left	0 0 6	250000	3V090

Shift by 10^3

Instruction 8 0 0 4

	Register	Decimal	Sexadecimal
Before Decimal Shift Left	0 0 6	123456789	75TUV15
After Decimal Shift Left	0 0 6	3197704712	TW991S08

The last example shows how the two high order positions affect the shifted decimal number. Instead of obtaining 456789000, the decimal result 3197704712 was obtained in FA 6.

Command	Decimal Shift Right (Divide by 10^1 to 10^8)
Code	8 8 - -
Instruction	8 8 P N PN = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6

Description

The decimal shift right command divides the contents of FA 6 (accumulator) by the power of ten specified by the PN tetrads of the instruction. The powers of ten range from one to eight. The result of the division appears in FA 6. If the result of this division exceeds the 32 bit capacity of FA 6, the overflow is lost off the low order end of FA 6 (TO-1).

Although the decimal shift right command is actually a binary division by powers of ten, it produces results that are the equivalent of a decimal right shift.

Examples

Shift by 10^{-5}

	Register	Decimal	Sexadecimal
Instruction	8 8 1 0		
Before Decimal Shift Right	0 0 6	1234567	12V687
After Decimal Shift Right	0 0 6	12	00000U

Shift by 10^{-1}

Instruction	8 8 0 1		
Before Decimal Shift Right	0 0 6	123	7T
After Decimal Shift Right	0 0 6	12	U

Command	Binary Shift Left (multiply by 2^1 to 2^8)
Code	9 0 - -
Instruction	9 0 P N PN = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6

Description

The binary shift left command shifts the contents of FA 6 (accumulator) to the left the number of binary positions specified by the PN tetrads of the instruction. The number of positions range from one to eight. The result of the shift appears in FA 6. Zeros are inserted into the low order positions vacated. If the result of the shifting exceeds the 32-bit capacity of FA 6, the overflow is lost off the high order end (T7-8 end). This command is equivalent to multiplying FA 6 by powers of two from 2^1 to 2^8.

Example

Shift Left by 2^4

	Register	Decimal	Sexadecimal
Instruction	9 0 0 8		
Before Binary Shift Left	0 0 6	54	00000036
After Binary Shift Left	0 0 6	864	00000360

Shift Left by 2^8

Instruction	9 0 8 0		
Before Binary Shift Left	0 0 6	872394770	33XXT012
After Binary Shift Left	0 0 6	4289729024	XXT01200

Command	Binary Shift Right (divide by 2^1 to 2^8)
Code	9 8 - -
Instruction	9 8 P N PN = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6

Description

The binary shift right command shifts the contents of FA 6 (accumulator) to the right the number of binary positions specified by the PN tetrads of the instruction. The number of positions can range from one to eight. Zeros are inserted in the high order positions that the data has vacated. The result of the shift appears in FA 6. If the result of the shifting exceeds the 32 bit capacity of FA 6 (accumulator), the overflow is lost off the low order end (T0-1 end). This command is equivalent to dividing FA 6 by powers of two from 2^1 to 2^8.

Example

Shift Right by 2^1

	Register	Decimal	Sexadecimal
Instruction	9 8 0 1		
Before Shift Right	0 0 6	3	00000003
After Shift Right	0 0 6	1	00000001

Shift Right by 2^6

Instruction	9 8 2 0		
Before Shift Right	0 0 6	805306368	30000000
After Shift Right	0 0 6	12582912	00U00000

Command	Binary Left End Around Shift
Code	8 U - -
Instruction	8 U P N PN = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6, 0 0 5

Description

The binary left end around shift command shifts the contents of FA 6 (accumulator) and FA 5 to the left the number of binary positions specified by the PN tetrads of the instruction. The number of positions can range from one to eight. The data shifted off the high order (T7-8) end of FA 6 is inserted into the positions vacated in the low order (T0-1) end of FA 5; the data shifted off the high order (T7-8) end of FA 5 is inserted into the positions vacated in the low order (T0-1) end of FA 6.

This command is a circular shift where no information is lost.

Example

End Around Shift 8

Instruction	8 U 8 0	
	Register	**Sexadecimal**
Before End Around Shift	0 0 6	01234567
	0 0 5	80123456
After End Around Shift	0 0 6	23456780
	0 0 5	12345601

Command	Binary Shift Right Maintain High Order Bit
Code	9 U - -
Instruction	9 U P N PN = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6

Description

The binary shift right maintain high order bit command shifts the contents of FA 6 (accumulator) to the right the number of binary places specified by the PN tetrads of the instruction. The number of positions can range from one to eight. The result of the shift appears in FA 6. Zeros are inserted into the high order positions vacated if the high order (T7-8) bit position was a zero; ones are inserted into the high order positions vacated if the high order (T7-8) bit position was a one. If the result of the shifting exceeds the 32 bit capacity of FA 6, the overflow is lost off the low order (T0-1) end.

This command preserves the high bits in shifting right and so can be used with either positive or negative numbers.

Example

	Register	Sexadecimal
Shift Right 4 Maintain High Order		
Instruction	9 U 0 8	
Before Maintain Right Shift	0 0 6	01234567
After Maintain Right Shift	0 0 6	00123456
Shift Right 4 Maintain High Order		
Instruction	9 U 0 8	
Before Maintain Right Shift	0 0 6	81234567
After Maintain Right Shift	0 0 6	X8123456

INTERVENTION INTERROGATE

The Monrobot Mark XI has eight control panel switches which can be interrogated by the program to determine whether the switches are set or not set. These switches can serve as manual program interrupt controls or as program break points. Testing a switch to determine its condition requires two Monrobot XI instructions. One instruction is the conditional jump described previously; the other instruction is the intervention interrogate instruction which is described below.

Each switch has a two tetrad address. Table 5 gives the sexadecimal address coding for each switch. In describing the switch address in this manual, the symbols AD will be used for switch address.

Switch Number	Sexadecimal Address	
	T1	TO
1	0	1
2	0	2
3	0	4
4	0	8
5	1	0
6	2	0
7	4	0
8	8	0

Interrogate Switch Coding

Table 5

More than one switch can be addressed for interrogation by the intervention interrogate instruction. The address for the switches becomes the sexadecimal sum of the different switch addresses used. For example, the address for switch numbers 2, 6, and 8 is S2.

Switch Number	Sexadecimal Address
2	0 2
6	2 0
8	8 0

Address 2, 6, 8 = S2

Command Intervention Interrogate

Code U 4 - -

Instruction U 4 A D AD = 01, 02, 04, 08, 10, 20, 40, 80

Fast Access Registers

0 0 6

Description

The intervention interrogate instruction loads the 32 bit positions of fast access register 6 (accumulator) with ones if the addressed switch is set; it loads the 32-bit positions with zeros if the addressed switch is not set. If more than one switch is addressed by the instruction, every switch addressed must be set to obtain ones in the accumulator.

This instruction should be followed by a conditional jump instruction if the programmer wants the program to branch on the condition of the switch.

Example

Test Switch 2, Switch Set

Instruction	U 4 0 2	
	Register	**Sexadecimal**
Before Intervention Interrogate	0 0 6	00125671
After Intervention Interrogate	0 0 6	XXXXXXXX

Test Switch 6, Switch not set

Instruction	U 4 2 0	
Before Intervention Interrogate	0 0 6	301XW203
After Intervention Interrogate	0 0 6	00000000

SPECIAL COMMANDS

The Monrobot Mark XI has special commands to set FA 6 and FA 5 to zero and to set FA 6 to all ones or minus one. These are described in this section. Other special commands exist to generate certain constants which may be useful to the programmer. A list of these commands, constants which they generate, and the rules for these commands are located in Appendix 2.

Another special command is a do-nothing command which allows the program to step without affecting any registers.

Command	Clear FA 6 (Accumulator)
Code	U 5 0 0
Instruction	U 5 0 0

Fast Access Registers

0 0 6

Description

The clear FA 6 command sets the 32 bit positions of fast access register 6 (accumulator) to zero.

Example

Clear FA 6

Instruction	U 5 0 0		
	Register	Decimal	Sexadecimal
Before Clear FA 6	0 0 6	125	7V
After Clear FA 6	0 0 6	0	0

Command Clear FA 5

Code S 0 0 0 also $07X

Instruction S 0 0 0

Fast Access Registers

0 0 5

Description

The clear FA 5 instruction sets the 32 bit positions of fast access register 5 to all zeros.

Example

Clear FA 5

Instruction S 0 0 0

	Register	Decimal	Sexadecimal
Before Clear FA 5	0 0 5	801	321
After Clear FA 5	0 0 5	0	0

Command	Set FA 6 to all Ones
Code	U 4 0 0
Instruction	U 4 0 0

Fast Access Registers

0 0 6

Description

The set FA 6 to all ones command will set the 32 bit positions of fast access register 6 to all ones. This effectively places a negative one in the register, since all ones in two's complement is minus one.

Example

Set FA 6 to all ones

Instruction U 4 0 0

	Register	Decimal	Sexadecimal
Before Set FA 6 to -1	0 0 6	0	00000000
After Set FA 6 to -1	0 0 6	-1	XXXXXXXX

Command No Operation

Code S 1 0 0

Instruction S 1 0 0

Fast Access Registers

None

Description

The no operation command does nothing to the program except advance the program to the next step. It is used as a fill instruction in either the A step or B step.

OTHER NO-OPS ARE:

0200	0700	V006
0300	6400	W400
0600	7400	

64CC Where any numbers can be used in place of the 'CC' to create still a no-op i.e 6401, 64XX.

Command Stop

Code 0 0 - -

Instruction 0 0 C C C C = any 2 tetrad character

Fast Access Registers

None

Description

The stop instruction halts all computer operations. Data can be written in the T0 and T1 tetrads to identify the stop command.

CONTROL PANEL

The Monrobot Mark XI control panel allows for entry of boot-strap programs, contains indicator lights, the intervention switches, and the necessary control switches to turn the computer power on and off. Figure 13 shows the control panel face.

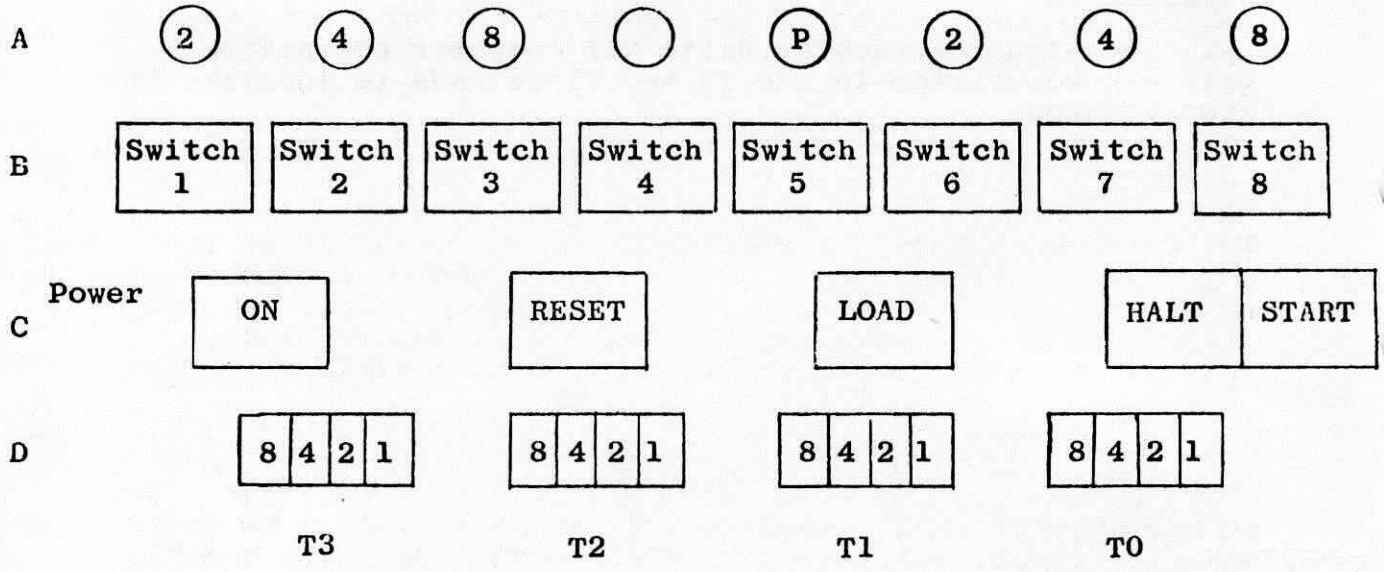


Figure 13

The functions of each switch on the panel are described below:

Row A is the input-output lights.

Row B is the intervention interrogate switches.

Row C is the control switches.

Row D is the control register indicators.

Control Switches

Each control switch is a two-position switch. The position of the switch whether on or off is indicated by the switch face being lit if on or unlit if off.

ON Switch

The ON switch controls the power to the computer. If it is lit, the computer power is on. The initial depression of the ON switch turns the computer on and clears the control register and instruction register to zero. It takes approximately one minute from the time the ON switch is depressed before the computer is ready to process data.

RESET Switch

The RESET switch controls the automatic operation of the computer. Any time the switch light is on, automatic operations will not occur. This switch can be depressed at any time during computer operation and computations will cease and the control loop will be set to zero. Consequently, this switch should be used with discretion.

LOAD Switch

The LOAD switch controls the transfer of data from fast access register 6 (accumulator) to the instruction register when the reset switch is lit. This switch provides for the entry of program data to the computer when automatic operations have ceased. Depression of this switch has no effect on computer operations if the reset switch is not lit.

HALT Switch

The HALT switch limits the automatic operation of the computer to one instruction. If the HALT light is on, the computer will process the instruction in the control register and stop. Depression of this switch when the reset switch light is on has no effect on computer operations. The HALT switch's effect upon computer operations is canceled by depressing the switch when the HALT light is on.

START Switch

The START switch initiates automatic computer operation. Depression of the START switch causes the computer to do the next instruction in the instruction register if the reset light is on; if the halt light is on, the next instruction in the control register will be performed. The START switch light being on will indicate that the computer is in automatic operation. Further depression of the START switch will have no effect on computer operation. The START switch light will remain on until automatic computer operations are interrupted by the depression of the reset switch, halt switch, or the use of the stop command in the program.

Control Register Lights

The control register lights give visual indication of the contents of the control register if the halt switch is lit. The control register will show the next instruction that will be performed when the start switch is depressed.

As Figure 13 shows, the control register is composed of 16 lights arranged in four groups. Each group represents a tetrad in the instruction. A light is on when the bit position of the instruction contains a one; it is off when the bit position contains a zero. Thus the instruction corresponds exactly to the instruction as written by the programmer except for the following four cases:

(1) Conditional jumps which are unsuccessful have the T2-4 bit lit in addition to the instruction lights.

(2) Stop instructions have the T2-1 light lit.

(3) Typewriter input has the T2-1 light on while awaiting an input.

(4) The binary left end around shift has the T2-2 light on if FA 5 has a one in the T7-8 position prior to the shift.

Input-Output Lights

The top row of indicators on the control panel are the input-output lights. The three extreme left-hand lights are for the three input devices; the three extreme right-hand lights are for the three output devices. The light next to the output lights is for parity indication. The eighth light is a spare and does not have any use at present.

Input Lights

The input lights come on whenever the device associated with the light is addressed by the program and no character is available for input at that address. The light remains on until a character is entered at that device.

Output Lights

The output lights come on whenever a device is addressed by an output command and the output cannot be made to the device. Reasons why the output cannot be made are that the device is busy, i.e., the device has not finished with the previous character sent to the device, or that the device is not available at the address specified. If more than one device is addressed, the lights for all devices addressed will remain on until the output instruction to all devices addressed has been accomplished.

Parity Light

The parity light comes on when the last character entered had an even number of bits and an input device has been addressed. The light will remain on until a character has been entered at the addressed input device. The parity light will not come on if a parity error occurs and an output command is given prior to the command for reading an input device. The manner in which the parity light functions gives the programmer the option to light the light and indicate parity error or to ignore the parity light and indicate parity error in some other manner. It also allows the programmer to ignore parity errors for characters which do not have parity such as five-channel characters.

Intervention Interrogate Switches

The eight switches in the middle of the control panel are the intervention interrogate switches. They are numbered from left to right as shown in Figure 13. Each switch has an address by which the computer can address it. Figure 14 gives each switch and its address.

Switch	Address
1	0 1
2	0 2
3	0 4
4	0 8
5	1 0
6	2 0
7	4 0
8	8 0

Figure 14

The switches are two position. When the switch light is on, the switch will load FA 6 with ones when addressed; when the switch light is off, the switch will load FA 6 with zeros when addressed.

The numbering of the switches is for programming clarification only. The numbers are not on the switch face, and the switches can have any writing on them that is desirable.

Reset Entry

There are two modes of input to the Monrobot Mark XI computer. These modes are program input and reset input. Program input was described under the input command and occurs when the computer is running automatically. Reset input is nonprogrammed, fixed format and can only occur when the computer is in nonautomatic operation. This occurs when the reset light has been turned on either by a depression of the reset switch or the computer program having a stop command.

Reset input is necessary to load initial (boot-strap) programs and to start the computer at the desired program step when automatic operation has ceased.

Reset input is possible only through the typewriter or sixteen-key keyboard which is located at input address 2 (device 1). Consequently, this input address must always have either one of these two devices at this input address for boot-strapping and for starting the computer. Reset entry input has sixteen legitimate characters. These are the tetrads 0 to 9 and S to X. Each character in this mode of entry is one tetrad or four bits. The characters are entered into the T0 position of FA 6 (accumulator). Every time a character is entered, the contents of FA 6 are shifted four binary places to the left and the character entered is inserted into the cleared T0 position of FA 6 (accumulator). Entering eight characters will fill FA 6; however, if more than eight are entered, only the last eight entered will remain in FA 6. The characters prior to the last eight are lost off the high order end.

If the typewriter is used as the reset input entry device, any character entered other than the legal sixteen tetrads will create errors in FA 6. If this illegal entry is made, all characters entered prior to the illegal character must be re-entered.

Starting Automatic Program

To place the computer in automatic operation when the reset light is on, a jump instruction must be entered into the control loop. This jump instruction will contain the address of the first instruction register in the sequence and allow automatic operation to occur after this jump.

The jump instruction is entered by reset entry into fast access 6 (accumulator). Depressing the load switch transfers the jump instruction to the instruction register. Depressing the start switch transfers the jump instruction into the control register where automatic operation starts. The computer now runs automatically according to the program until either the reset switch is used or a stop command is programmed. It is possible to start automatic operation after a stop command without reset entry by depressing the start switch providing the reset switch has not been operated.

Boot-Strap Techniques

When the computer is first used, it will not have any programs stored on the drum. The method of getting data onto the drum in this condition is called boot-strapping. Boot-strapping involves using the reset entry, load, and start switches. This method also can be used to check-out programs, change the contents of registers, or observe the contents of registers.

Boot-strapping involves loading the control loop with a store instruction that will transfer the contents of FA 6 to the register specified by the store instruction. This operation is a two-step one. The first step is reset entry into FA 6 of the store instruction. Figure 15 shows the form that this instruction should have. Step A of the register is the store instruction.

T7	T6	T5	T4	T3	T2	T1	T0
T	A	D	R	0	0	0	0

Figure 15

When the load switch is depressed, the contents of FA 6 are transferred to the instruction register, leaving FA 6 unchanged.

The second step is reset entry into FA 6 of the instruction that is to be boot-strapped into storage to serve as program steps or data. These instructions replace the store instruction in FA 6. Once the eight tetrads of the two steps are entered, the start switch is depressed. The A step of the instruction register is put into the control register where the computer interprets it as a transfer of the contents of FA 6 to the specified address. When this transfer is accomplished, the next step (the B step) is interpreted by the control register; and since it is a stop command, automatic operation stops. As many of these two-step boot-strap operations are done as are necessary to load the boot-strap program. When the entire program is loaded, the computer is put into automatic operation by entering a jump command containing the starting address of the sequence in the first step and just depressing the start switch in the second step.

Figure 15 gives a sample boot-strap program.

This same two-step operation is used whenever it is desired to change the contents of the register.

This boot-strap program will first be loaded into storage as a two-step operation. It then will be used to load the rest of memory automatically as a one-step operation. These two methods show in addition to boot-strapping the two methods of automatic starting when the reset light is on.

Boot-Strap

This program will be loaded into registers 0 0 7 to 0 0 S. It then will be used to load program data starting with register 1 0 0. The reset light must be on for the boot-strap program.

Enter	T0070000	Depress Load Switch
Enter	0000T100	Depress Start Switch
Enter	T0080000	Depress Load Switch
Enter	V007X00S	Depress Start Switch
Enter	T0090000	Depress Load Switch
Enter	T0073007	Depress Start Switch
Enter	T00S0000	Depress Load Switch
Enter	00000001	Depress Start Switch

Entry of those four program registers gives the program in memory as shown in Figure 15.

Once the program has been entered, it may be started by entering the unconditional jump command with 0 0 7 as its address and pressing the load and start switches. The program is now in automatic operation. The first command is a stop command which places the Monrobot Mark XI in reset entry. The eight tetrads of register 1 0 0 are entered into FA 6

MONROBOT MARK XI PROGRAM SHEET		PAGE
PROGRAM Reset Entry Program	PROGRAMMER	DATE

REGISTER			STEP	CONTENTS				NOTES
		0	A					
			B					
		1	A					
			B					
		2	A					
			B					
		3	A					
			B					
		4	A					
			B					
		5	A					
			B					
		6	A					
			B					
0	0	7	A	0	0	0	0	Stop--allow for reset entry; press start switch.
			B	T	1	0	0	Store in storage register.
0	0	8	A	V	0	0	7	Instruction 7 $\longrightarrow$ FA 6.
			B	X	0	0	S	Add 00000001 to modify address.
0	0	9	A	T	0	0	7	Return modified instruction to register 0 0 7.
			B	3	0	0	7	Jump to 0 0 7 to await next entry.
0	0	S	A	0	0	0	0	Constant one to modify address.
			B	0	0	0	1	
		T	A					
			B					
		U	A					
			B					
		V	A					
			B					
		W	A					
			B					
		X	A					
			B					

MO-92

Figure 15

and the start switch is depressed. The computer advances the program one step to store the data in register 1 0 0. It then transfers register 0 0 7 to FA 6, adds one to modify the storage address for the next register in sequence, and restores the modified instruction in 0 0 7. It then jumps to 0 0 7 to obtain the next register contents. This will continue until all the program instructions in this sequence are stored.

Computer Aid to Program Checking

The Monrobot Mark XI has machine features for aiding the programmer in detecting errors in his program.

Foremost among these aids is the ability to execute one instruction and stop. This one instruction operation is the halt switch, which is located on the control panel. When the halt switch has been used and the halt light is on, the computer will do the instruction that is shown by the instruction lights when the start switch is operated. In this manner the programmer can manually step through the program and check it for correct sequencing.

The computer has also a fast access display box. When this box is connected to an oscilloscope, the programmer can have displayed for his use the contents of the instruction register, fast access 6 (accumulator), fast access 5, and fast access 4. The box has a five-position switch labeled T31, CL, ACC, MP, MC. The switch can be placed at any position or moved from one position to another to show the contents of the desired register. The T31 position is used to synchronize the oscilloscope. Using this fast access display box, the halt switch, and the control register lights, the programmer will be able to check the progress of the program. The control loop and instruction register display instructions. FA 6, FA 5, and FA 4 give all that is needed to know about the numerical data that enters into operation. All information displayed is in binary form.

The contents of any register can be brought to the oscilloscope for display when the computer is in the reset mode. The programmer enters via reset entry a load command giving the address of the register which he wishes to have displayed. This instruction has the form of V A D R 0 0 0 0. Depressing the load and start switches will load the contents of the addressed register into FA 6 where it will be displayed on the oscilloscope. The contents of any register can be changed when the computer is in the reset mode. The programmer enters via reset entry a store command giving the address of the register he wishes to change. This instruction has the form T A D R 0 0 0 0. This is transferred to the instruction register by the load switch. The data that is to be entered into the addressed register is now entered via reset entry and the start switch is depressed. The data entered now replaces the data that was in the addressed register.

OPERATION TIMING

Monrobot Mark XI has sixteen sector times per drum revolution in which to execute instructions. Every instruction takes at least four sector times. Instructions that have reference to general storage have additional time called access time; some commands require more than four sector times for execution. Appendix I gives the execution and access time for every instruction in Monrobot Mark XI.

As illustrated and explained in the section on the control instructions, Monrobot Mark XI executes two instructions and then the automatic jump loads the instruction register with the next two instructions. The automatic jump instruction then has its address augmented by one. The new instruction address is in the same sector of the drum for sixteen consecutive jumps and then the sector is increased by one. Consequently, the computer cannot select a general storage instruction register more than once per drum revolution when operating sequentially because a full drum revolution is required before the addressed sector is available again. This means that sixteen sector times are available in every drum revolution for executing two instructions and the automatic jump. It also means that this is the maximum number of instructions that will be executed when operating sequentially through the computer storage registers. (When instructions are executed from fast access registers, four instructions may be executed during a drum revolution because there is no access time.) However, this maximum is not always accomplished because execution times and access times may exceed the sixteen sector times available per drum revolution. In order that three instructions may be executed per drum revolution whenever possible, minimum access coding should be used.

Minimum Access Coding

The purpose of minimum access coding is to make it possible for the computer to process three instructions per drum revolution by locating operand registers and instructions so that they will be available with minimum access. Of the sixteen sector times in each revolution, twelve are necessary for execution of the three instructions. This leaves an excess of four sector times which can be divided among the three instructions. These sector times can be used to spread the range of registers which the instruction may have access to or they can be used as execution times for instructions that take longer than four sector times for execution such as multiple shifts.

One Drum Revolution Operations

The rules for obtaining two instructions and the automatic jump per drum revolution when either one or both instructions have operands in general storage are as follows:

N = sector address of automatic jump instruction

A = sector address of A program step

B = sector address of B program step.

Case I (one drum revolution)

The sector address of the A program step can lie between

$N + 4$ and $N + 8$

or $N + 4 \le A \le N + 8$ modulo 16.

The sector address of the B program step can lie between

$A + 4$ and $A + 8$

or $A + 4 \le B \le A + 8$ modulo 16.

The sector address of the B program step cannot exceed

$N - 4$

or $B + 4 \le N$ modulo 16.

N modulo 16 is the remainder after the sector is divided by 16.

Example

N Sector	A Step Sector(s)	B Step Sector(s)
3	7	T, U, V, W, X
	7, 8	U, V, W, X
	7, 8, 9	V, W, X
	7, 8, 9, S	W, X
	7, 8, 9, S, T	X

The example shows how the sectors used in A affect the range of B and vice versa.

Minimum Access for More Than One Drum Revolution

The following gives the rules for determining the drum revolutions used when the operands exceed the range required for one drum revolution per pair of instructions.

Case II (two drum revolutions)

The two instructions will take two drum revolutions when

$A = N + 9$, B can be any sector

or $B = N + 7$, A can be any sector

and also when either Case I or Case III does not apply.

Case III (three drum revolutions)

$N + 10 \leq A \leq N + 15$

and $N + 13 \leq B \leq N + 2$

and $14 \leq B - A \leq 3$

or $N \leq A \leq N + 3$

and $N + 13 \leq B \leq N + 6$

and $10 \leq B - A \leq 3$.

If the A step or the B step or both are instructions which do not refer to storage, the same rules apply except that A and B are computed in terms of sector times rather than sector addresses. These nonaddressable commands are the shift commands. The number of shifts made will determine whether the two instructions can be accomplished in a drum revolution. Instructions which require more sector times than can be accomplished in a drum revolution such as multiply and detract are not optimized. After a multiply or detract, optimization begins with the next instructions that can be accomplished in a drum revolution.

INPUT-OUTPUT TIMING

The basic input-output operation speed is four sector times. The limitation in the number of inputs and outputs that can be made is the nature of the device from which either input or output occurs. Inputs or outputs cannot be made at a rate faster than the devices will accept characters. If the commands are executed before the device is ready, a condition called busy occurs. This condition causes computer operation to suspend until the device is ready to accept the character. Where possible, program steps should be scheduled so that there is no computer waiting time to either input or output.

Device	Operation Time
Tape Reader	50 milliseconds
Tape Punch	50 milliseconds
Typewriter	100 milliseconds
16-key Keyboard	100 milliseconds
Teletypewriter	100 milliseconds.
Card Reader	62.5 milliseconds
Card Punch	62.5 milliseconds

APPENDIX I

Command and Access Time

Command	Operation Time	Access Time
Detract	7 + 2n	0 - 15
Multiply	38	0 - 15
Store	4	0 - 15
Interchange	4	None
Load	4	0 - 15
Subtract	4	0 - 15
Add	4	0 - 15
Extract	4	0 - 15
Jump	4	0 - 15
Jump Mark	4	0 - 15
Jump Zero	4	0 - 15
Jump High Order 1	4	0 - 15
Input	4	0 if available
Output (FA 5 and instruction)	4	0 if available
Multiply by 10	4 + (m - 1)	None
Divide by 10	4 + 2(m - 1)	None
Binary Shift Left	4 + (m - 1)	None
Binary Shift Right	4 + (m - 1)	None
Binary End Around	4 + 2(m - 1)	None
Binary Right Shift Maintain High Order	4 + (m - 1)	None
Intervention Interrogate	4	None
Clear FA 6	4	0
Clear FA 5	4	0

Command	Operation Time	Access Time
Set FA 6	4	0
No Operation	4	0
Stop	2	0

n = number of subtractions

m = number of shifts.

APPENDIX II

Constant Generation

The Monrobot Mark XI can generate certain numbers in one instruction. These numbers can usually be used as constants in the program. In the generation of these numbers, FA 6 must always be cleared to zero prior to using a constant generate instruction. If FA 6 is clear as the result of a previous operation, this method of obtaining a constant can prove economical in both storage space and time. Each constant is generated by a special form of a shift command. Each instruction constant generator is given below with both its decimal and sexadecimal equivalents.

Accumulator must be clear

Constants Generated in One Instruction

Instruction	Decimal	Sexadecimal
—9401	—1	1
8201	2	2
8301	3	3
—9404	4	4
8401	5	5
8501	6	6
8601	7	7
8701	8	8
8102	—10	S
—9410	16	10
8202	20	14
8302	30	1W
9420	32	20
8402	50	32
8502	60	3U
—9440	64	40
8602	70	46
8702	80	50
8104	—100	64
—9480	128	80
8204	200	U8
8304	300	12U
8404	500	1X4
8504	600	258

Instruction	Decimal	Sexadecimal
8604	700	2TU
8704	800	320
8108	1 000	3W8
8208	2 000	7V0
8308	3 000	TT8
8408	5 000	1388
8508	6 000	1770
8608	7 000	1T58
8708	8 000	1X40
8110	10 000	2710
8210	20 000	4W20
8310	30 000	7530
8410	50 000	U350
8510	60 000	WS60
8610	70 000	1 1170
8710	80 000	1 3880
8120	100 000	1 86S0
8220	200 000	3 0V40
8320	300 000	4 93W0
8420	500 000	7 S120
8520	600 000	9 27U0
8620	700 000	S SW60
8720	800 000	U 3500
8140	1 000 000	X 4240
8240	2 000 000	1W 8480
8340	3 000 000	2V U6U0

Instruction	Decimal	Sexadecimal
8440	5 000 000	4U 4T40
8540	6 000 000	5T 8V80
8640	7 000 000	6S UXU0
8740	8 000 000	7S 1200
8180	10 000 000	98 9680
8280	20 000 000	131 2V00
8380	30 000 000	1U9 U380
8480	50 000 000	2XS X080
8580	60 000 000	393 8700
8680	70 000 000	42U 1V80
8780	80 000 000	4U4 T400

APPENDIX III

TABLE OF POWERS OF 2

n	2^n	n	2^n
0	1	32	4 294 967 296
1	2	33	8 589 934 592
2	4	34	17 179 869 184
3	8	35	34 359 738 368
4	16 x	36	68 719 476 736
5	32	37	137 438 953 472
6	64	38	274 877 906 944
7	128	39	549 755 813 888
8	256 x	40	1 099 511 627 776
9	512	41	2 199 023 255 552
10	1 024	42	4 398 046 511 104
11	2 048	43	8 796 093 022 208
12	4 096 x	44	17 592 186 044 416
13	8 192	45	35 184 372 088 832
14	16 384	46	70 368 744 177 664
15	32 768	47	140 737 488 355 328
16	65 536 x	48	281 474 976 710 656
17	131 072	49	562 949 953 421 312
18	262 144	50	1 125 899 906 842 624
19	524 288	51	2 251 799 813 685 248
20	1 048 576 x	52	4 503 599 627 370 496
21	2 097 152	53	9 007 199 254 740 992
22	4 194 304	54	18 014 398 509 481 984
23	8 388 608	55	36 028 797 018 963 968
24	16 777 216	56	72 057 594 037 927 936
25	33 554 432	57	144 115 188 075 855 872
26	67 108 864	58	288 230 376 151 711 744
27	134 217 728	59	576 460 752 303 423 488
28	268 435 456	60	1 152 921 504 606 846 976
29	536 870 912		
30	1 073 741 824		
31	2 147 483 648		

1 2 3 4 5 6 7 8 9 S T U V X

10 \ 1 units 1.25

10 \ 1 unit 1000

Character	Eight-Level Code EL X O P 8 4 2 1	Monrobot XI Sexadecimal Input-Output Code	Instruction Output Code
TAB	0 0 1 1 1 1 1 0	1W	5W
BACK SPACE	0 0 1 1 1 1 0 1	1V	5V
CARRIAGE RETURN	1 0 0 0 0 0 0 0	80	80
UPPER CASE	0 1 1 1 1 1 0 0	3U	7U
LOWER CASE	0 1 1 1 1 0 1 0	3S	7S
TAPE FEED			$500

APPENDIX IV

Binary to Decimal and Decimal to Binary Conversion

Every decimal number can be represented as a digit times a power of ten. For example, 1073 is equivalent to $1 \times 10^3 + 0 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$. To convert such numbers to binary, the simplest method would be to use the binary equivalents of the decimal values. To convert binary numbers to decimal numbers, it would be necessary to find the decimal equivalents of the binary values. Monrobot Mark XI has special commands to facilitate these conversions.

For example, to convert the decimal number 32 to its binary equivalent 100000, the following operations would be done. Both three and two would be given their binary equivalents: 3 = 0011 and 2 = 0010. 3 would be multiplied by the binary equivalent of decimal 10 (1010), giving the binary value of 11110 or decimal 30. The binary equivalent of decimal 2 (0010) would be added to 11110, giving binary 100000 or decimal 32.

In converting from binary to decimal, the inverse would be done. It would be necessary to find how many powers of 10 were in each binary number. The simplest method is to subtract from the binary number the binary equivalent of the power of 10 as many times as possible and use this result as the digit for that power of 10. The remainder can then be divided by the next lowest power of 10 for the next digit. This process is continued until the number is converted. For example, decimal 256 = 100000000 in binary.

```
    Decimal 100 =        1100100     binary

    Decimal 10 =            1010     binary

    Decimal 1 =                1     binary

    Subtracting        100000000

                         1100100     (1)
                       ---------
                        10011100

                         1100100     (2)
                       ---------
                          111000
```

Subtracted twice gives 2 x 100. The remainder is then divided by 10:

```
111000
  1010   (1)
------
101110
  1010   (2)
------
100100
  1010   (3)
------
 11010
  1010   (4)
------
 10000
  1010   (5)
------
   110
```

This subtraction is done five times or 5 x 10. The remainder is 110 or divided by 1 binary gives decimal 6. The result of the conversion is 256.

All conversions in Monrobot Mark XI can be done in this fashion. The commands multiply and divide by powers of 10 give facility in shifting binary values by their decimal equivalents. The command detract will subtract the powers of 10 and give the count of the number of subtractions.

The following pages give examples of subroutines for doing conversion in the computer using these methods.

Input

The routine shown here assumes that the input is coming from a sixteen-key keyboard whose code is

Decimal Value	Binary Value
0	0000
1	0001
2	0010
3	0011
4	0100

APPENDIX V

2,048 Word Drum Address Structure

The 2,048 word drum is available as an optional feature of the Monrobot XI computer to provide twice the storage of the 1,024 word drum basic computer.

Monrobot XI programs written for the 1,024 word basic computer are compatible with the higher capacity drum. The basic 1,024 register addresses are from 000 to 3XX in both models of the computer.

An 8's bit in tetrad 2 indicates a register address in the additional 1,024 word storage area. These register addresses are consecutive from 800 to TXX. The computer will not sequence automatically from register 3XX to register 800.

There are no alterations or exception cases of the Monrobot XI command structure as described in the Program Manual when applied to the 1,024 word additional storage computer.

Examples of coding:

Command	Basic 1,024	Additional 1,024
Add	X050	X850
Jump mark	35XX	3VXX

Basic 1,024 register addresses:

	Sexadecimal				Binary											
					T2				T1				T0			
Tetrad	T2	T1	T0	Power of 2	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0
		↓				↓				↓				↓		
	3	X	X		0	0	1	1	1	1	1	1	1	1	1	1

Additional 1,024 register addresses:

Sexadecimal			Binary											
2	1	0	11	10	9	8	7	6	5	4	3	2	1	0
8	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	↓			↓				↓				↓		
T	X	X	1	0	1	1	1	1	1	1	1	1	1	1