MINUTEMAN COMPUTER USERS GROUP

THE TULANE UNIVERSITY OF LOUISIANA

NON SIBI SED SUIS

1834

D17B COMPUTER DOCUMENTATION

MINUTEMAN COMPUTER USERS GROUP

D17B COMPUTER PROGRAMMING MANUAL

Edited By

Charles H. Beck

Professor and Director

SYSTEMS LABORATORY

Department of Electrical Engineering

School of Engineering

TULANE UNIVERSITY

REPORT MCUG-4-71

* Programming Features
* Instruction Set
* Programming Techniques
* Programming Examples

SEPTEMBER 1971

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

PROGRAMMING FEATURES OF THE D17B

The D17B is a small serial-binary general-purpose computer which was designed as an airborne control computer. It has several features which make machine language programming different than for other general-purpose computers.

The D17B has a disc memory which rotates at 6000 r/min. The disc contains 21 concentric programmable channels which are addressable by even octal numbers from 00 through 50. Each channel contains 128 sectors or words, numbered octally from 00 through 177. Each word contains 24 bits for the purpose of storing data or instructions. There are a total of 2727 addressable words on the disc. Since the disc rotates through 128 sectors in 0.01 second, the length of time required to record, write, or pass over any one sector is $0.01/128 \doteq 78.2\ \mu s$. This interval defines one word time.

The serial presentation of information on the rotating disc necessitates careful coding of a program to insure the minimum access time between instructions as well as between an instruction and the data on which it operates. This scheme of coding is referred to as minimal delay coding (MDC).

In addition to an Accumulator (A) and a Lower Accumulator (L), there are rapid-access storage loops, each of which contains less than 128 words. These loops (U, F, E, and H) have lengths of 1, 4, 8, and 16 words, respectively. Information stored in the U, F, E, and H loops is available after 1, 4, 4, and 8 word times respectively, rather than after one entire disc revolution. Significant characteristics of these loops are tabulated in Table 1 along with the two input loops, V and R.

The D17B has a "flag store" mode which provides for simultaneously storing the previous contents of the Accumulator in certain specified channels coincident with the execution of an instruction. This feature eliminates the need for an additional instruction to perform this frequent operation. The D17B can also be used to perform parallel- or multi-processing such as the simultaneous execution of two identical single-precision arithmetic operations. This not only provides for the execution of two operations during one word time, but also effectively doubles the memory available for data storage. In single-precision data storage each word is divided into two 11-bit words at the sacrifice of some precision.

WORD LENGTH	CHANNEL	FLAG CODE (S_F)	DESCRIPTION
-	-	00	No Flag
4	52	02	F-loop
1	-	04	Telemetry
128	50	06	Channel 50
8	56	10	E-loop
1	64	12	L-loop (Lower Accummulator)
16	54	14	H-loop
1	60	16	U-loop
4	72	-	R-loop
4	70	-	V-loop

Table 1. Flag store codes and memory specifications.

CHAPTER 2

PROGRAMMING THE D17B

2.1 Addressing

Each word location on the disc memory is specified by a channel and sector number denoted (c,s) for channel, sector. For example, (10, 113) would specify the word located in sector 113 of channel 10. Twelve bits are required for direct addressing all information stored in memory; five are used for channel (since channels are only specified by even octal numbers) and seven for the sector.

2.2 Word Formats

Four word formats are available: double-precision fixed or full-word data, single-precision fixed or split-word data, unflagged instructions, and flagged instructions. The formats are shown in Figure 1.

The full-word format accommodates 23 binary bits in bit positions T_1 through T_{23}. Bit position T_{24} contains the sign bit which indicates a negative number when equal to 1 and a positive number when equal to 0. When a negative number is entered into memory, the two's complement of the magnitude of the number must be entered in bit position T_1 through T_{23}.

The split-word format provides for the storing of two 10-bit numbers and a sign bit for each. The "right-half" number is stored in bit position T_1 through T_{11} and the "left-half" number in positions T_{14} through T_{24}. Bit positions T_{12} and T_{13} are unused and their values are not predictable.

The fixed binary point is located between T_{24}-T_{23} for the full-word data format; the two fixed binary points for the split-word format are located between T_{24}-T_{23} and T_{11}-T_{10}. However, if it is necessary for the programmer

SIGN	WHOLE NUMBER																						
T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_{9}	T_{8}	T_{7}	T_{6}	T_{5}	T_{4}	T_{3}	T_{2}	T_{1}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	7			7			7			7			7			7			7			7	

SIGN	10 BIT BINARY FRACTION									X	X	SIGN	10 BIT BINARY FRACTION										
T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_{9}	T_{8}	T_{7}	T_{6}	T_{5}	T_{4}	T_{3}	T_{2}	T_{1}
1	1	1	1	1	1	1	1	1	1	1	X	X	1	1	1	1	1	1	1	1	1	1	1
	7			7			7			6			3			7			7			7	

Figure 1. Instruction and data word formats. (Sheet 1 of 3)

DECIMAL NUMBER	BINARY REPRESENTATION	OCTAL EQUIV.
+1	*******	*****
	0.11111····11	37777777
	0.11111····10	37777776
•	•	•
•	•	•
•	•	•
	0.00000····01	00000001
0	0.00000····00	00000000
	1.11111····11	77777777
•	•	•
•	•	•
•	•	•
	1.00000 10	40000002
	1.00000 01	40000001
-1	*******	*****

Note: Numbers with magnitude of +1 to -1 may be represented in binary as shown in the table above. All negative numbers are shown in the 2's complement form as they are represented in the D17B.

Figure 1. Instruction and data word formats. (Sheet 2 of 3)

UNFLAGGED INSTRUCTION (T_{20} = 0)

T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1
OPERATION CODE				FLAG	NEXT INSTRUCTION SECTOR ADDRESS							CHANNEL NUMBER					SECTOR NUMBER						
OP				F	S_P							C					S						

FLAGGED INSTRUCTION (T_{20} = 1)

T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1
OPERATION CODE				FLAG	FLAG STORAGE LOCATION			SECTOR OF NEXT INSTRUCTION				CHANNEL NUMBER					SECTOR NUMBER						
OP				F	S_F			S_P				C					S						

Figure 1. Instruction and data word formats. (Sheet 3 of 3)

to assume a location for the binary point he must appropriately allow for it in programming and in interpreting results.

The unflagged instruction format consists of four parts: a 4-bit operation code (OP), a 1-bit flag (F), the 7-bit sector address (Sp) of the next instruction within the active memory channel, and the 12-bit channel and sector address (c,s) of the operand. Operation codes are two-digit quad-octal numbers which determine the instruction type. A complete explanation of available instructions is given in Chapter 5.

For an unflagged instruction, F is always 0. Since Sp contains no channel specification, the next instruction must always be located in the same channel as the one preceeding it. This restriction does not apply to the transfer (TRA) and transfer on minus (TMI) instructions which have different formats. The rightmost twelve bits of these transfer instructions indicate the address of the next specified instruction which may be located in any channel and sector of memory.

A flagged instruction is identical to an unflagged instruction except that the flag bit is 1 and Sp is reduced from seven to four bits. The remaining three bits are used for the flag store code (Sf). When the flag bit is 1, the contents of the Accumulator will be stored in the channel specified by Sf during the first word time of execution of the instruction. Determination of the storage location is explained in section 4 of this chapter. Channel 50, the L, U, E, F, and H loops, and the telemetry output channel are available as flag storage locations. The channel addresses and flag store codes are listed in Table 1. Because Sp is limited to only four bits for specification of the sector address of the next instruction, the next instruction after a flagged instruction will come from one of the next 16 sectors after operand agreement (beginning of execution). Therefore, Sp is used to indicate only the four least significant bits of the sector address of the next instruction.

2.3 Quasi-Octal Notation

Although words are represented internally by 24 binary bits, for programming convenience an octal representation is used as input to reduce the number of digits which need to be punched. However, since the instruction word formats are not compatible with the eight 3-bit groupings of bits needed for octal notation, an intermediate or "quasi-octal" notation is used. Figure 2 shows the quasi-octal grouping of bits for the two instruction formats. Pseudo-zero bits are inserted as shown and the quasi-octal representation may then be found by grouping each set of three bits and finding the octal equivalents. The octal representation may then be found by grouping and adding the quasi-octal numbers as shown. Since it is easier to work with octal digits than with binary when writing programs, all numbers which refer to the contents of parts of the two instruction formats will be represented in quasi-octal notation so that the total word may then be conveniently converted to an 8-digit octal number.

2.4 Minimal Delay Coding

Because information on the memory disc is transferred serially, the location of successive instructions and their operands on the disc is extremely important. Five steps are necessary to complete an instruction; Figure 3 shows how the D17B can simultaneously complete certain steps of successive instructions. With minimal delay coding an instruction can be completed in the number of word times equal to the execution time of that instruction. Therefore, to execute a number of sequential instructions in the minimum number of word times, the instructions should be separated by

UNFLAGGED INSTRUCTION

OP CODE				F	NEXT INSTRUCTION SECTOR							OPERAND												
T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_{9}	T_{8}	T_{7}	T_{6}	T_{5}	T_{4}	T_{3}	T_{2}	T_{1}	
OP				F	S_P							C					S							
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BINARY
7			4	0	1	7			7			7			6		1	7			7			QUASI-OCTAL
7			5			7			7			7			7			7			7			MAXIMUM OCTAL VALUE

FLAGGED INSTRUCTION

OP CODE				F	NEXT INSTRUCTION SECTOR							OPERAND												
T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_{9}	T_{8}	T_{7}	T_{6}	T_{5}	T_{4}	T_{3}	T_{2}	T_{1}	
OP				F	S_F			S_P				C					S							
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BINARY
7			4	2	1	6		1	7			7			6		1	7			7			QUASI-OCTAL
7			7			7			7			7			7			7			7			MAXIMUM OCTAL VALUE

Figure 2. Quasi-octal instruction coding.

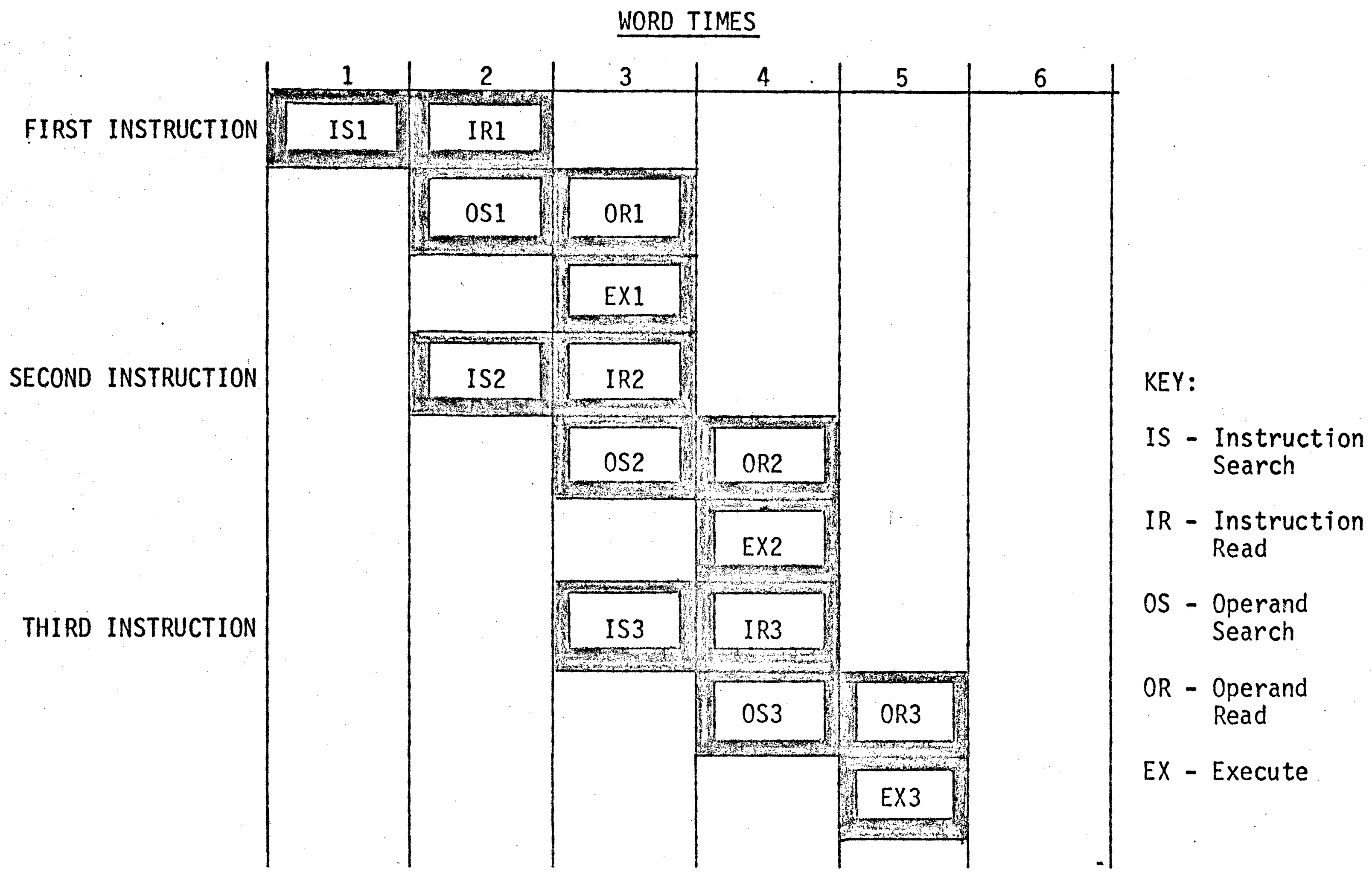

Figure 3. Steps required for instruction completion.

n-1 sectors where n is the execution time, measured in word times, of the first instruction of each pair of instructions considered. However, it is permissible to locate an instruction in a sector which is alloted for the execution of the previous instruction.

For example, if a CLA instruction is followed by an ADD instruction, the ADD instruction should be located in the sector following the CLA instruction, since the execution time of the CLA instruction is one word time. If a MPY instruction is followed by an ADD instruction, the ADD instruction should be located 13 sectors after the MPY instruction since the execution time of the MPY instruction is 13 word times. However, if the 13 sectors following the MPY instruction are used for the next instruction of the program, a delay of one disc revolution will take place between the end of execution of the MPY instruction and the next instruction.

Figure 3 also shows that, for a given instruction, the operand can be read during the word time immediately following the reading of the instruction. Therefore, when writing a program, the operand of each instruction should be located in the sector following the sector location of the instruction. In order to maintain the desired instruction locations, the operands should be located in other channels in the correct sector since corresponding sectors in all channels are available during the same word time. It is usually convenient to locate all operands in the same channel.

If it is necessary to write on (or read from) a sector which is not located optimally in relation to the instruction, the next instruction should be located a sufficient number of sectors later so that, when the word is stored (or read), the operand sector or some later sector will contain the next instruction. If this is not done the disc may have to rotate almost an entire revolution to reach the sector of the next instruction. For example, if an ADD instruction is located in sector 71 and specifies that the number to be added is in sector 75, then the next instruction should be located in sector 75.

The E and H loops each have an intermediate read head in addition to the normal read head to enable more rapid access to stored information. For example, if a word was stored in sector 3 of the H-loop, which has 16 sectors, it would not be available for 16 word times with the normal read head. However, the intermediate read head makes the word available after 8 word times in sector 13.

2.5 Store Operations

Words may be stored in any channel except one word loops using the STO instructions if the Enable Write switch is ON. If it is OFF, only channel 50 and the F, E, and H loops may be written on. When a STO instruction is encountered, the sector storage address is compared with the addresses recorded on the sector track of the disc. When the two addresses agree, the contents of the Accumulator are stored in the word which has a sector address of two less than that specified since the write head is located two sector addresses behind the read head.

Therefore, for STO instruction to be executed in one word time the address to be stored should be specified at one sector greater than the sector location of the instruction. The word will then be stored in the sector location preceding the instruction. The channel selected for storage should be different from the channel in which the instructions are located in order not to interfere with their optimal location. However, the channel selected for storage may be the same as the channel where the instructions are located so that instruction modification may be performed under program control. For a flag store to channel 50, the word stored will be automatically placed in sector s-2, where s is the sector address of the operand.

Flag storing to the rapid-access loops is done in a different manner. In a flagged instruction, if the flag-code of one of the rapid-access loops is placed in Sf, then the previous contents of the Accumulator will be stored in the sector of the designated loop which corresponds to the least significant bits of the operand sector address.

For example, flag storing in the E-loop which contains eight words by an instruction located in sector 116, whose operand is located in sector 117, will cause the previous contents of the Accumulator to be stored in word seven of the E-loop.

2.6 Fine Countdown

In real-time programming, timing control is an important factor. The execution time of some particular part of a program may vary for different situations. The fine countdown mode, one of the special features of the D17B computer, can be utilized for program timing control when it is necessary. Once the program transfers to a certain subroutine, the fine countdown mode can be initiated simultaneously. Fine countdown is performed independently of the program in memory.

When the fine countdown mode is entered, the 0, +1, and -1 incremental inputs to V, instead of accumulating in their respective sectors of V, cause the product of the input and the contents of their sector to be algebraically added to the contents of the U-loop. The U-loop, assuming that it contained a positive number by flag storage originally, is counted down at a rate depending upon the sign and magnitude of the numbers placed in the V-loop and upon the sign and frequency of the inputs. When the contents of the U-loop become negative, D_5 is turned on independently by ${}_1D_5$ = Fc Ux T_{24}, and a discrete output signal (D_{16}) is issued automatically. The discrete register must be set to zero prior to entering the fine countdown mode, and the third bit of phase register, P_3, must be set to 0. During fine countdown ($F_c = 1$), the V-loop recirculates or receives new information via the STO instruction. The special discrete output indicates a solution of the equation:

$$0 = K_u + (K_o + K_2) I_1 + K_1 I_2 + K_3 I_3$$

where K_u is the contents of the U-loop before fine countdown. K_o through K_3 are the constants in the V-loop, and I_1 through I_3 are the incremental inputs. I_1 will be sampled once every two word times whereas I_2 and I_3 will be sampled once every four word times.

If K_o through K_3 are stored as 1 and I_1 through I_3 are made to be -1 by setting V_k equal to 1 and V_s equal to 1, the fine countdown mode will last K_u + 2 word times until a discrete signal is generated. If the execution of the subroutine is completed before the fine countdown terminates, the computer can be made to idle until the discrete signal transfers the program to an appropriate location. In this fashion, the programmer can assure that the computer will require equal execution time, K_u + 2 word times, for a certain subroutine and gain the ability of timing control. A typical example utilizing fine countdown mode can be found in Sec. 4.6.

Chapter 3

THE INSTRUCTION SET AND PROGRAMMING TECHNIQUES

Included at the end of this manual is a list and a brief explanation of all the available instructions, their mnemonics, quad-octal codes, execution times, and other pertinent data. It is the purpose of this chapter to explain the uses and limitations of this instruction set as well as the special features of the D17B (i.e. flag storing, rapid-access loops, and split-word operations).

3.1 Arithmetic Operations

Instructions for adding, subtracting, multiplying, complementing, and storing are available. Separate instructions are used for adding, subtracting, and multiplying with split-word formats. These instructions are basically the same as in other general-purpose machines, with the exception of the Multiply Modified (MMP) and Split Multiply Modified (SMM) instructions which may have their operand addresses modified as determined by the exclusive-OR of the three least significant bits of the operand and the contents of the phase register.

Addition, subtraction, and complementation each require one word time for execution. Multiplication requires 13 word times and split multiplication 7 word times. When storage space and time are at a premium it is often desirable to use the split-word instructions, especially when high precision is not needed. Split words are accurate to a tenth of one per cent.

Noticeably absent from this set is a division instruction. The D17B does not have a hardware division capability since it is a relatively time consuming process. To divide by a constant, the reciprocal of the number is stored and multiplication may then be used. To divide by a power of two, right shifting is used.

It is often necessary to use the appropriate Taylor series expansion to compute functions such as SIN, EXP, etc. These expansions contain sums of terms which are very similar. To program these expansions with the minimum number of operations needed, a technique called "nesting" is used.

For example, the expansion of EXP (x) is:

$$\mathrm{Exp}(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdot \cdot \cdot$$

After nesting, the first five terms are:

$$\mathrm{Exp}(x) = K_1 + x(K_1 + x(K_2 + x(K_3 + x(K_4))))$$

where: $K_1 = 1$

$K_2 = 1/2! = 0.5000$

$K_3 = 1/3! = 0.1666$

$K_4 = 1/4! = 0.0416$

The programs for this and other functions are given in Chapter 4.

3.2 Shifting Operations and Masking

Shifting instructions are available to move the contents of the Accumulator either to the left or right. There are separate instructions available for the full-word format and for each word of the split-word format.

These instructions are briefly described in the list of instructions in Chapter 5. Using a left-shift instruction causes the contents of the Accumulator to be shifted to the left the specified number of bits. The most significant bits are lost and zeros fill the vacated positions of the least significant bits. In performing a right-shift the least significant bits are lost and zeros fill the vacated positions of the most significant bits, except when the number is negative; in this case ones fill the vacated bit positions.

A shift instruction requires a minimum of two word times to be executed, specifically for a zero- and a one-bit shift. Otherwise, it requires one word time more than the number of shifts desired (n + 1 word times).

An instruction which is commonly used in conjunction with shifting operations is the ANA instruction or "logical AND to Accumulator." The ANA instruction replaces the contents of the Accumulator with the "logical AND" between corresponding bits of the Accumulator and the Lower Accumulator. The ANA instruction can be used to perform masking, and in conjunction with shifting operations it may be used to perform packing and unpacking of words of less than 24 bits (e.g. 11-bit split-words).

In large general-purpose machines, a "logical OR to Accumulator" is available and is used primarily for packing while the ANA instruction is used for unpacking. It is therefore appropriate to consider how the ANA instruction may be used for both operations and, at the same time, demonstrate how flag storing may be used to reduce execution time.

To unpack, for example, suppose that three 8-bit words are stored in a word in memory, and the middle word is to be right justified in the Accumulator, as is usually required. A "mask" containing 1's in bits 9 through 16 and zeros in all other bit positions is first placed in the Accumulator (using CLA). The entire word to be unpacked is then placed in the Accumulator (using CLA), and at the same time the "mask" is flag-stored in the Lower Accumulator. The ANA instruction is then used, leaving the bits in bit positions 9 through 16 unchanged and all other bits zero. A right-shift of eight bits will then right-justify the 8-bit word.

Packing requires two more word times. For example, if it is desired to repack the 8-bit word which was unpacked in the previous example, the word is first placed in the Accumulator and then shifted left eight bit positions. A word containing zeros in bits 9 through 16 and ones in all other bit positions is then added to the Accumulator, resulting in the word to be packed to be located in bits 9-16 and ones in all other bit positions. The word containing the other two 8-bit words is then placed in the Accumulator, while simultaneously flag-storing the previous contents of the Accumulator in the Lower Accumulator. Assuming bits 9-16 of the Accumulator are zero (if they are not they must be masked to zero), a word containing ones in bits 9-16 and zeros in all other bit positions is added to the Accumulator, resulting in all ones in bits 9-16. Use of the ANA operation will then produce the original word (containing three 8-bit words) in the Accumulator.

3.3 Control Operations

The basic control operations available are Unconditional Transfer (TRA), Transfer on Minus (TMI), and Halt and Proceed (HPR). Now, although this is a small set compared to those of larger general-purpose computers, it can be used to effect all other control operations.

To achieve a transfer on plus, the two's complement of the word is taken before testing it with the TMI instruction. A transfer on zero may be achieved by first using the MIM instruction to make the number negative. Then, by using the TMI instruction, the contents of Sp will specify the next instruction if the Accumulator is zero.

Another way to change the use of the TMI instruction is by subtracting a one from the word to be tested in the least significant bit position. This has the effect of changing the use of the TMI instruction to transfer on minus or zero to one location and plus to another, rather than on minus to one location and on plus or zero to another. Making this change, rather than simply complementing the word and then testing it, allows freedom of channel location if the contents of the Accumulator are zero. The MIM and COM instructions may be used together to obtain the absolute value of a number.

This is illustrated in Chapter 4 as a programming example.

3.4 Input/Output Operations

The program, composed of instruction and data words, is entered into memory either through the manual control board, flexowriter, teletypewriter, paper tape reader, card reader, or magnetic tape reader. Instruction and data characters can be read in during the load/verify mode; sequential memory locations are assumed unless a location control character is present.

The main instruction which causes the D17B to output the contents of the Accumulator to an external source is character output (COA). To effect transmission of a single character the four MSB's from the Accumulator are loaded into a special register. As this register is loaded, a parity bit is generated. After loading the register and generating the parity bit, a timing pulse is available, and the four bits plus the parity bit are transmitted out in parallel. Continuing in this fashion it would require six COA instructions to transmit an entire computer word.

By properly shifting right once and then using COA, the full octal contents of the Accumulator can be transmitted by using eight COA instructions. Output devices currently being used are the light display, flexowriter, teletypewriter, and paper tape punch.

There are several additional instructions which enable the programmer to enter or extract data from the D17B under program control. Discrete input lines (X1c thru X24c and Y1c thru Y24c) are available for entering 24-bit data words. Among them X20c thru X24c are inputs derived from Fc, Dr, P3, P1, and P2 flip-flops respectively. The true level for the discrete input lines is -10V, and 0 V represents the false level.

Discrete output instructions provide for outputting a level on one of 28 discrete output lines. Only one line can be energized at a time with the execption of D01, D02, D03, and D04. D04 can be activated with one of D01, D02, or D03, thus allowing for two lines at most to be energized at a time. Furthermore, D10 and D21 are ANDed with Dr, the gyro bottoming detector. True and false output level for DOA are -25V and +10V respectively. Under program control three voltage outputs can be updated one at a time from the computer. The analog voltage outputs are proportional to the digital information held in buffer registers which are associated with the D/A converters. Split word data is used to originate the analog signal. The eight MSB's of either the right or left half word are fed into one of the three buffer registers. The total voltage swing is approximately ±20V.

Three binary outputs are available on two lines for each output from the $(G)_{1-3}$ flip-flops and are of a pulse nature. Execution of a BOA, BOB, or BOC will cause the eight MSB's of the Accumulator to be increased or decreased by one depending on the existing state of the associated G flip-flop. At the completion of the instruction the sign position of the Accumulator will determine the new setting of the associated G flip-flop. The true output level is -10V, and the false level is -1V.

Chapter 4

PROGRAMMING EXAMPLES

4.1 Exp (x)

This program computes e^x for any given value of x, using the following algorithm:

$$\mathrm{Exp}(x) \doteq 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$$

$$\doteq K_1 + x(K_1 + x(K_2 + x(K_3 + x(K_4))))$$

$$\text{Where } K_1 = 1.0,\ K_3 = 0.16666,$$

$$K_2 = 0.5,\ K_4 = 0.04167$$

Symbols such as K_1, K_2, etc. are used to represent the constants and designations U, L, etc. to represent the rapid-access loops to provide more flexibility in application and better understanding. These programs are written using MDC; therefore, a programming sheet has been included as an example, for the first program, to show actual memory locations for a program which starts in sector 000 of channel 2.

It is assumed that X is in the U-loop and Exp(x) will be placed in the Accumulator (A-loop).

SECTOR LOCATION	OP	S_F	S_p	C,S	COMMENTS
000	CLA		001	K_4	$(K_4) \longrightarrow (A)$
001	MPY		016	U	$x(K_4)$
016	ADD		017	K_3	$K_3 + x(K_4)$
017	MPY		034	U	$x(K_3 + x(K_4))$
034	ADD		035	K_2	$K_2 + x(K_3 + x(K_4))$
035	MPY		052	U	$x(K_2 + x(K_3 + x(K_4)))$
052	ADD		053	K_1	$K_1 + x(K_2 + x(K_3 + x(K_4)))$
053	MPY		070	U	$x(K_1 + x(K_2 + x(K_3 + x(K_4))))$
070	ADD		071	K_1	$K_1 + x(K_1 + x(K_2 + x(K_3 + x(K_4))))$

PROGRAMMER: ________ TITLE: Exp (x)

TULANE UNIVERSITY -- Electrical Engineering
MINUTEMAN D17B COMPUTER PROGRAM -- SYMBOLIC CODE

DATE: / /
PAGE: 1

LOCATION	INSTRUCTION				
CH 02	OP	Sf	Sp	CH/Sec	COMMENTS
00	44	0	01	0401	(K4) → (A)
01	24	0	16	6000	X · (K4)
02					
03					
04					
05					
06					
07					
10					
11					
12					
13					
14					
15					
16	64	0	17	0417	K3 + XK4
17	24	0	34	6000	X(K3+XK4)
20					
21					
22					
23					
24					
25					
26					
27					
30					
31					
32					
33					
34	64	0	35	0435	K2+X(K3+XK4)
35	24	0	52	6000	X(K2+X(K3+XK4))
36					
37					

LOCATION	INSTRUCTION				
CH 02	OP	Sf	Sp	CH/Sec	COMMENTS
40					
41					
42					
43					
44					
45					
46					
47					
50					
51					
52	64	0	53	0453	K1+X(K2+X(K3+XK4))
53	24	0	70	6000	X(K1+X(K2+X(K3+XK4)))
54					
55					
56					
57					
60					
61					
62					
63					
64					
65					
66					
67					
70	64	0	71	0471	Exp (x)
71					

CONSTANT LOCATIONS

CONSTANT	CHANNEL/SECTOR LOCATION
x	60/000
K_1	04/053, 071
K_2	04/035
K_3	04/017
K_4	04/001

4.2 Absolute Value

This program computes $|x|$ for any given value of x by using the MIM and COM instructions. The binary representation of x is initially in the U-loop and $|x|$ is found in the Accumulator.

SECTOR LOCATION	OP	S_F	S_P	C,S	COMMENTS
000	CLA		001	U	$(x) \longrightarrow (A)$
001	MIM		002		$-\ |x|$
002	COM		003		$|x|$

4.3 Unpacking

This program unpacks the middle split-word from a word containing three 8-bit split-words. The word to be unpacked is in the U-loop and the unpacked, right-justified split-word is found in the Accumulator.

SECTOR LOCATION	OP	S_F	S_P	C,S	COMMENTS
000	CLA		001	M_1	$(M_1) \longrightarrow (A)$
001	CLA	L	002	U	$(A) \longrightarrow (L)$, $(U) \longrightarrow (A)$
002	ANA		003	42,s	The mask is used to unpack the desired word.
003	ARS		014	32,10	(A) are shifted right by eight bit positions.

M_1 = 00000000 11111111 00000000 (located in sector 1)

4.4 Packing

This program packs an 8-bit word (X) in bits 9-16 of the U-loop. Two other 8-bit words are located in bits 1-8 and 17-24 and zeros are initially in bits 9-16.

SECTOR LOCATION	OP	S_F	S_P	c,s	COMMENTS
000	CLA		001	X	$(X) \longrightarrow (A)$
001	ALS		012	22,010	(A) are shifted left eight bits
012	ADD		013	W_1	$(A) + (W_1) \longrightarrow (A)$
013	CLA	L	014	U	$(A) \longrightarrow (L)$, $(U) \longrightarrow (A)$
014	ADD		015	W_2	$(A) + (W_2) \longrightarrow (A)$
015	ANA		016	42,	The three 8-bit words are packed as required.

W_1 = 11111111 00000000 11111111 (located in sector 13)

W_2 = 00000000 11111111 00000000 (located in sector 15)

4.5 Subroutine Linkage

A linkage between the main program and the subroutine is essential when various subroutines are used. Before transferring to the location of the subroutine, two preparations have to be made. First, the data must be stored into appropriate memory locations. Second, a returning transfer instruction must be stored at the end of the subroutine. The result will be located either in the Accumulator or in a specific location as specified by the given subroutine. The subroutine can be used as many times as desired if a proper linkage is established. Following is an example illustrating this technique.

MAIN PROGRAM			SUBROUTINE	
LOCATION	INSTRUCTION		LOCATION	INSTRUCTION
M-1	Last instruction before linkage		L	First instruction of subroutine
M	CLA	MM1	.	. . . .
M+1	STO	LL1		
M+2	CLA	MM2		
M+3	STO	LL2		
.	...	...		
A	CLA	MM		
A+1	STO	N+1	.	. . . .
A+2	TRA	L	N	Last instruction of subroutine
A+3	First instruction after linkage		N+1	TRA A+3
.	...	...		
MM1	DATA 1		LL1	DATA 1
MM2	DATA 2		LL2	DATA 2
.	... .		.	... .
MM	TRA	A+3	LL	RESULT

Some examples of subroutines that have been written and executed on the D17B in the Systems Laboratory at Tulane University are listed below.

10-bit Binary to 12-bit BCD

8-character COA Subroutine

Von Neumann Division

Waveform Generation

4.6 Fine Countdown Mode (FCM)

In order to keep track of program timing, fine countdown can be used to equalize the execution time of a subroutine for different situations. Usually, the fine countdown mode will last as long as the maximum execution time of the specific subroutine. Then, K_u should be the maximum execution time minus two expressed in wdt. This quantity is flag stored before entering fine countdown.

A typical program is shown below.

MAIN PROGRAM				SUBROUTINE		
LOCATION		INSTRUCTION		LOCATION	INSTRUCTION	
L-1		Last instruction before FCM		N	First instruction of subroutine	
L		LPR	7400	•	• • •	• • •
L+1		CLA	LL1			
L+2	F,U	CLA	LL2	M	Last instruction of subroutine	
L+3		STO	7000			
L+4		STO	7001	NN	DIB	$D_{16} \longrightarrow A_{24}$
L+5		STO	7002	NN+1	TMI	Go to NN+2 if -ve, NN if ≥ 0
L+6		STO	7003			
L+7		EFC	62,s	NN+2	DOA	2600
L+8		TRA	N	NN+3	TRA	L+9
L+9		HFC	60,s			
L+10		First instruction after FCM				
•••		••••	••••			
LL1		"K_u"				
LL2		"1"				

4.7 D-A and A-D Conversion

This program will show some of the special I/O capabilities of the D17B. A digital number will be converted to an analog voltage level. This signal will be transmitted to the input of an A-D converter, and the resulting digitized equivalent will be returned to the D17B using the Discrete Inputs.

Sector Location	OP	S_F	S_P	c,s	COMMENTS
000	LPR		001	7200	Set P_1
001	CLA		002	X	(X) $\longrightarrow$ (A)
002	VOA		003	3000	$(A)_{24-17} \longrightarrow VO_{11}$, send analog voltage to A-D converter.
003	DOA		004	2611	-25V at D_{09} to pulse A-D.
004	DOA		005	2600	10V at D_{09} to pulse A-D.
005	DIB		006	5000	$(Y_1-Y_{24}) \longrightarrow (A)$ Return converted digital value to (A).
006	HPR		007	2200	Halt and compare converted value with (X).

4.8 COA Subroutine (8-Octal Characters)

This program will send out eight octal characters through the COA output which can be interfaced with a peripheral I/O device.

SECTOR LOCATION	OP	S_F	S_P	c,s	COMMENTS
000	CLA		001	4601	(Data in 4601) $\longrightarrow$ (A)
001	CLA	L	002	4602	(A) $\longrightarrow$ (L) ("1" in 4602) $\longrightarrow$ (A)
002	ANA		003	4200	(A) $\cdot$ (L) $\longrightarrow$ (A)
003	ALS		007	2203	Left shift 3-bits
007	STO		010	4610	Store masked LSB in 4606.
010	CLA		011	4601	(Data in 4601) $\longrightarrow$ (A)

COA Subroutine (Continued)

011	ARS		013	3201	Right shift 1-bit
013	CLA	L	14	4614	(A) $\longrightarrow$ (L) 4614 (37777777) (A)
014	ANA		015	4200	(A)$\cdot$(L) $\longrightarrow$ (A) Make sign bit zero.
015	COA		017	4001	Character output $(A)_{24-21}$
017	ADD		020	4606	Add masked LSB.
020	STO		021	4603	(A) $\longrightarrow$ 4601 (Data)
021	CLA		022	4622	(4622) $\longrightarrow$ (A)
022	TMI		023	4430	Is 8th COA completed?
023	SUB		024	4624	(A)-4624(1) $\longrightarrow$ (A)
024	STO		025	4624	(A) $\longrightarrow$ 4622
025	CLA		026	4626	4626 $\longrightarrow$ (A), Delay counter
026	SUB		027	4627	(A)-4627(1) $\longrightarrow$ (A)
027	TMI		026	4400	Test for end of delay.
030	CLA		031	4631	4631(6) $\longrightarrow$ (A)
031	STO		032	4624	(A) $\longrightarrow$ 4622
032	HPR		033	2200	Halt

LOCATION	DATA
4601	Data for output
4602	00000001
4606	Storage - masked LSB
4614	37777777
4622	00000006
4624	00000001
4626	00000005
4627	00000001
4631	00000006

Chapter 5

LIST OF INSTRUCTIONS

This section contains a list of the available instructions with a brief explanation of each. The following abbreviations are used to facilitate a more concise listing:

( ) ≡ The contents of (subscripts a-b refer to bit positions a thru b).

$\longrightarrow$ ≡ replaces

wdt ≡ word times

(c,s) = (m) ≡ contents of operand bit positions

Other symbols have been explained previously.

5.1 Arithmetic Operations

CLA	Clear and Add $(m) \longrightarrow (A)$	44	c,s	1 wdt
STO	Store Accumulator $(A) \longrightarrow (m)$ except if: (m) is a one word loop, or a cold channel and the enable write switch is in the disable position in which case (A) and (m) are unchanged. If (c) is 50, F, H, or E, only L may be used for flag storing.	54	c,s	1 wdt
ADD	Add $(m) + (A) \longrightarrow (A)$	64	c,s	1 wdt
SAD	Split Add $(m)_{14-24} + (A)_{14-24} \longrightarrow (A)_{14-24}$ and $(m)_{1-11} + (A)_{1-11} \longrightarrow (A)_{1-11}$	60	c,s	1 wdt
SUB	Subtract $(A) - (m) \longrightarrow (A)$	74	c,s	1 wdt
SSU	Split Subtract $(A)_{14-24} - (m)_{14-24} \longrightarrow (A)_{14-24}$ and $(A)_{1-11} - (m)_{1-11} \longrightarrow (A)_{1-11}$	70	c,s	1 wdt
MPY	Multiply $(A) \longrightarrow (L)$ and $(A) \cdot (m) \longrightarrow (A)$	24	c,s	13 wdt
SMP	Split Multiply $(A)_{1-11} \longrightarrow (L)_{14-24}$ and $(A)_{14-24} \longrightarrow (L)_{1-11}$ $(A)_{14-24} \cdot (m)_{14-24} \longrightarrow (A)_{14-24}$ and $(A)_{1-11} \cdot (m)_{1-11} \longrightarrow (A)_{1-11}$	20	c,s	7 wdt

MMP	Multiply Modified	34	c,s	13 wdt

Execution is the same as for MPY; however, the operand channel address is modified before execution. Each of the three least significant bits of the operand channel address ($T_8 - T_{10}$) may be changed ($0 \longrightarrow 1$ or $1 \longrightarrow 0$) if the corresponding phase register bit ($P_1 - P_3$) is ONE. The operand channel bit remains unchanged if the corresponding phase register bit is ZERO. This is equivalent to the EXCLUSIVE OR of the operand bits and the phase register bits.

SMM	Split Multiply Modified	30	c,s	7 wdt

It is the same as SMP but with the conditions for MMP.

COM	Complement	40	46,s	1 wdt

The 2's complement of (A) $\longrightarrow$ (A), (s) are ignored.

MIM	Minus Magnitude	40	44,s	1 wdt

If (A) > 0, the 2's complement of (A) $\longrightarrow$ (A). (s) are ignored. If (A) $\leq$ 0, (A) are unchanged.

5.2 Shifting Operations

ALS	Accumulator Left Shift	00	22,s	s+1 wdt

(A) are shifted by $(s)_{1-5}$ bit positions.

ARS	Accumulator Right Shift	00	32,s	s+1 wdt

(A) are shifted right by $(s)_{1-5}$ bit positions.

SAL	Split Accumulator Left Shift	00	20,s	s+1 wdt

$(A)_{14-24}$ and $(A)_{1-11}$ are shifted left by $(s)_{1-5}$ bit positions.

SAR	Split Accumulator Right Shift	00	30,s	s+1 wdt

$(A)_{14-24}$ and $(A)_{1-11}$ are shifted right by $(s)_{1-5}$ bit positions.

SLL	Split Left Word Left Shift	00	24,s	s+1 wdt

$(A)_{14-24}$ are shifted left by $(s)_{1-5}$ bit positions. $(A)_{1-11}$ unchanged.

SLR	Split Left Word Right Shift	00	34,s	s+1 wdt

$(A)_{14-24}$ are shifted right by $(s)_{1-5}$ bit positions. $(A)_{1-11}$ unchanged.

SRL	Split Right Word Left Shift	00	26,s	s+1 wdt

$(A)_{1-11}$ are shifted left by $(s)_{1-5}$ bit positions. $(A)_{14-24}$ unchanged.

SRR	Split Right Word Right Shift	00	36,s	s+1 wdt

$(A)_{1-11}$ are shifted right by $(s)_{1-5}$ bit positions. $(A)_{14-24}$ unchanged.

5.3 Control Operations

TRA	Transfer	50	c,s	1 wdt

The next instruction is specified by (m); (Sp) are ignored and (c) cannot be 64, 70, or 72.

TMI	Transfer on Minus	10	c,s	1 wdt

If (A) < 0, the next instruction is specified by (m), (c) cannot be

64, 70, or 72 and all flag codes are defined. If (A) ≥ 0, the next instruction is specified by (Sp). Also, all flag codes are defined only if the TMI instruction is optimally located in relation to the last instruction, and the sector location is one less than (m).

HPR	Halt and Proceed	40	22,s	1 wdt

The Machine halts. The next instruction is specified by (Sp), which must specify the next sector after the HPR instruction when the computer is returned to the compute mode. (s) are ignored.

SCL	Split Compare and Limit	04	c,s	2 wdt

This instruction is defined only if the split words in (m) are positive or 0. The split words of (A) are compared simultaneously and independently with the corresponding parts of the contents of (m).

If $(A)_{24-14} > (m)_{24-14}$, $(m)_{24-14} \longrightarrow (A)_{24-14}$

If $(A)_{24-14} <$ 1's complement of $(m)_{24-14}$, 1's complement of $(m)_{24-14} \longrightarrow (A)_{24-14}$, otherwise no change. The same is true for $(A)_{11-1}$ and $(m)_{11-1}$.

ANA	Logical AND to Accumulator	40	42,s	1 wdt

(A) are logically ANDed, bit by bit, with (L) and the result stored in (A).

LPR	Load Phase Register	40	7-,s	1 wdt

Bits directly from $(c,s)_{8,9,5}$ of the instruction word will become $(P)_{1,2,3}$. (m) are unaffected. The instructions shown below will set the $(P)_{1,2,3}$ individually. They may be combined into one instruction to set any configuration.

$$40xx7200 \longrightarrow (P)_{1,2,3} = 100$$
$$40xx7400 \longrightarrow (P)_{1,2,3} = 010$$
$$40xx7020 \text{ thru } 40xx7037 \longrightarrow (P)_{1,2,3} = 001$$

EFC	Enter Fine Countdown	40	62,s	1 wdt

Enter the fine countdown mode. (s) are ignored.

HFC	Halt Fine Countdown	40	60,s	1 wdt

Halt the fine countdown mode. (s) are ignored.

RSD	Reset Detector	40	20,s	1 wdt

The detector is turned off. (s) are ignored.

5.4 Input and Output Operations

DIA	Discrete Input A	40	52,s	1 wdt

$(X)_{1-19} \longrightarrow (A)_{1-19}$, Fc (Fine Countdown) $\longrightarrow (A)_{20}$,

D_r (Detector) $\longrightarrow (A)_{21}$, $(P)_{3,1,2} \longrightarrow (A)_{22-24}$. (s) are ignored.

DIB	Discrete Input B	40	50,s	1 wdt

$(Y)_{1-24} \longrightarrow (A)_{1-24}$. (s) are ignored.

DOA	Discrete Output A	40	26,s	1 wdt

$(I)_{1-5} \longrightarrow (D)_{1-5}$. D matrix selects one of the 28 output lines, numbered D_{1c-4c} & D_{8c-31c}, corresponding to the BCD representation of $(D)_{1-5}$. The step signal remains energized until the next DOA instruction is executed. $(I)_{6,7}$ are ignored.

VOA	Voltage Output A	40	30,s	1 wdt

If $(s)_4$ is 0, $(A)_{17-24}$ are converted to a proportional voltage and sent to voltage output A. If $(s)_4$ is 1, $(A)_{4-11}$ are converted and sent to voltage output A. When $(P)_{1-3}$ is X01, terminal 1 is selected for the voltage outputs; X11 for terminal 2, X10 for terminal 3 and 100 for terminal 4. $(s)_{1-3,\ 5-7}$ are ignored.

VOB	Voltage Output B	40	32,s	1 wdt

Same instruction as VOA except voltage output B is used.

VOC	Voltage Output C	40	34,s	1 wdt

Same instruction as VOA except voltage output C is used.

BOA	Binary Output A	40	10,s	1 wdt

If the previous binary output was +1 or -1, a 1 is subtracted from or added to $(A)_{17-24}$. Binary output A is set to a +1 if $(A)_{24}$ is 0, or a -1 otherwise. (s) are ignored.

BOB	Binary Output B	40	12,s	1 wdt

Same instruction as BOA except binary output B is used.

BOC	Binary Output C	40	02,s	1 wdt

Same instruction as BOA except binary output C is used.

COA	Character Output A	00	40,s	s+1 wdt

$(A)_{21-24}$ become available to output equipment for s wdts. (A) are shifted left four positions. COA is not defined when s is 0. $(c)_5$ must be 1 for COA. $(c)_{4-1}$ are ignored. Therefore, (c) may be any even octal number from 40-76.

A list of the D17B instruction repertoire which summarizes all the available instructions, their mnemonics and quad-octal codes is given in the following table.

	CODE	DESCRIPTION	NUMERIC CODE
ARITHMETIC	CLA	CLEAR AND ADD	44 c,s
	STO	STORE ACCUMULATOR	54 c,s
	ADD	ADD	64 c,s
	SAD	SPLIT WORD ADD	60 c,s
	SUB	SUBTRACT	74 c,s
	SSU	SPLIT WORD SUBTRACT	70 c,s
	MPY	MULTIPLY	24 c,s
	SMP	SPLIT WORD MULTIPLY	20 c,s
	MPM	MULTIPLY MODIFIED	34 c,s
	SMM	SPLIT WORD MULTIPLY MODIFIED	30 c,s
	COM	COMPLEMENT	40 46,s
	MIM	MINUS MAGNITUDE	40 44,s
SHIFT	ALS	ACCUMULATOR LEFT SHIFT	00 22,s
	ARS	ACCUMULATOR RIGHT SHIFT	00 32,s
	SAL	SPLIT ACCUMULATOR LEFT SHIFT	00 20,s
	SAR	SPLIT ACCUMULATOR RIGHT SHIFT	00 30,s
	SLL	SPLIT LEFT WORD LEFT SHIFT	00 24,s
	SLR	SPLIT LEFT WORD RIGHT SHIFT	00 34,s
	SRL	SPLIT RIGHT WORD LEFT SHIFT	00 26,s
	SRR	SPLIT RIGHT WORD RIGHT SHIFT	00 36,s
CONTROL	TRA	TRANSFER	50 c,s
	TMI	TRANSFER ON MINUS	10 c,s
	HPR	HALT AND PROCEED	40 22,s
	SCL	SPLIT COMPARE AND LIMIT	04 c,s
	ANA	LOGICAL AND TO ACCUMULATOR	40 42,s
	LPR	LOAD PHASE REGISTER	40 7-,s
	EFC	ENTER FINE COUNTDOWN	40 62,s
	HFC	HALT FINE COUNTDOWN	40 60,s
	RSD	RESET DETECTOR	40 20,s
INPUT/OUTPUT	DIA	DISCRETE INPUT A	40 52,s
	DIB	DISCRETE INPUT B	40 50,s
	DOA	DISCRETE OUTPUT A	40 26,s
	VOA	VOLTAGE OUTPUT A	40 30,s
	VOB	VOLTAGE OUTPUT B	40 32,s
	VOC	VOLTAGE OUTPUT C	40 34,s
	BOA	BINARY OUTPUT A	40 10,s
	BOB	BINARY OUTPUT B	40 12,s
	BOC	BINARY OUTPUT C	40 02,s
	COA	CHARACTER OUTPUT A	00 40,s

Table 2. D17B computer instruction set.