8.0 MK3870 INSTRUCTION SET

8.1 INTRODUCTION

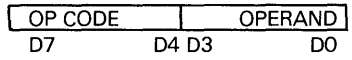
This section describes the execution and timing of the MK3870 Family instruction set. The 3870 instruction set is compatible with that of the multi-chip F8 Family.

8.2 3870 ADDRESSING MODES

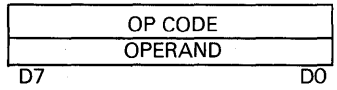
Most of the 3870 instructions operate on data stored in internal CPU registers, in main memory, in scratchpad memory, or in the I/O ports. The term "addressing mode" refers to how the address of this data is generated. This section gives a summary of the types of addressing used in the 3870.

8.2.1 IMMEDIATE ADDRESSING

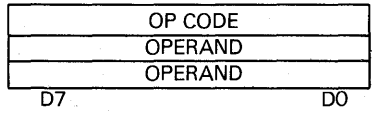
In this mode of addressing, the operand is contained in memory following the OP code of the instruction. Immediate addressing can be used in one byte, two byte, and three byte instructions. For a one byte instruction which uses immediate addressing the 3870 uses a special short form instruction format. The OP code is contained in the most significant four bits (D7-D4) of a byte in memory while a four bit operand is specified in the least significant four bits as shown below:



A two byte instruction which uses immediate addressing contains an 8-bit OP code which is specified in the first byte of the instruction and an 8-bit immediate operand in the second byte.

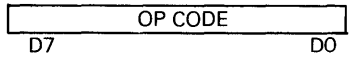


For certain instructions requiring greater than an 8 bit immediate operand, two bytes are required to specify an immediate operand. An example would be the DCI instruction, which loads an immediate value into the Data Counter. These instructions are three bytes long as shown below:



8.2.2 IMPLIED ADDRESSING

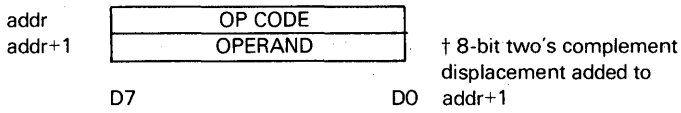
Instructions which use Implied addressing are instructions which are one byte long in which the operand or operands are implicitly specified in the instruction OP code itself. For example, an instruction which uses Implied addressing would be the 'LR Q,DC' instruction. The result of this instruction would be the contents of the Data Counter register loaded into the Q Linkage register.



8.2.3 RELATIVE ADDRESSING

Relative addressing is used exclusively by the Branch instructions in the 3870. Relative addressing uses one byte of data following the OP code to specify a displacement from the current Program Counter location to which a program jump can occur. This displacement is

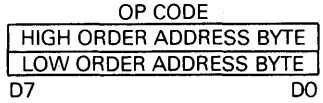
a signed two's complement number that is added to the address of the byte containing the displacement:



Relative addressing allows program jumps to locations within the range from -127 bytes backward to +128 bytes forward relative to the memory location containing the branch instruction op code. Relative addressing requires only two bytes of memory space to implement. For most programs relative branches are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements.

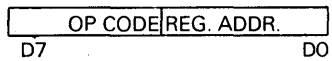
8.2.4 EXTENDED ADDRESSING

Extended addressing provides for up to two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where a subroutine may be called:



8.2.5 SCRATCHPAD ADDRESSING

Scratchpad addressing is utilized by instructions which access the Scratchpad Register Array. These instructions utilize the 3870's short form instruction format and are therefore only one byte long, with the instruction OP code specified in the most significant four bits:



The least sigificant four bits are used as an address to designate the scratchpad register whose contents are to be used as the operand in the instruction. The Scratchpad Register array contains 64 bytes of RAM. When the value of the four bit operand field is in the range of 0 through 0B (Hex) then the corresponding scratchpad location is accessed directly. When the operand value is 0C, 0D, or 0E (Hex), then the Indirect Scratchpad Register (IS) is used to point to the scratchpad location which is to be accessed. Note that the IS register can be used to point to any location in the scratchpad register array. Scratchpad locations 0CH, 0DH, 0EH, and 0FH are actually Linkage Registers K and Q, and, as such, may be directly accessed through instructions which use implied addressing.

The IS register is six bits wide which is divided into two halves, called IS Upper (ISU) and IS Lower (ISL). The Scratchpad registers may be thought of as an 8 x 8 array, with ISU pointing to a row of registers in the array and ISL pointing to a column of registers. This is illustrated below in Figure 8-1.

OCTAL REPRESENTATION OF SCRATCHPAD REGISTER ARRAY
Figure 8-1

		IS upper ->							
		0	1	2	3	4	5	6	7
	0	00	01	02	03	04	05	06	07
IS	1	10	11	12	13	14	15	16	17
lower	2	20	21	22	23	24	25	26	27
	3	30	31	32	33	34	35	36	37
	4	40	41	42	43	44	45	46	47
	5	50	51	52	53	54	55	56	57
	6	60	61	62	63	64	65	66	67
	7	70	71	72	73	74	75	76	77

This figure illustrates how the 2 octal digits of the IS register can be visualized as pointing to a scratchpad register in an 8 x 8 matrix.

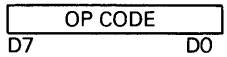
When the IS register is used to point to a register during the execution of an instruction which uses Scratchpad Addressing, the lower three bits of IS are modified according to the value specified in the operand as shown below:

OPERAND	IS LOWER
0CH	Unmodified
0DH	Incremented
0EH	Decremented
0FH	Illegal OP code

Thus, the lower half of the IS register can be automatically incremented, decremented, or left unmodified. When the IS is incremented or decremented during the execution of an instruction, only the lower half of the IS register is modified. As an example, suppose that the IS contains the octal value 27. If an instruction is executed which automatically increments the IS, the value will be changed to an octal 20 so that IS Upper is left unmodified.

8.2.6 INDIRECT MEMORY ADDRESSING

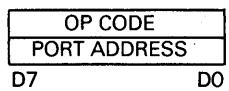
Instructions which operate on data contained in a location in Main Memory must access that data through the Data Counter Register (DC). These instructions are 1 byte in length. The one byte OP code specifies the implicit use of the Data Counter to perform an indirect access of main memory to fetch the 8-bit operand:



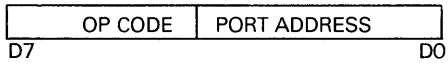
The value of the Data Counter is automatically incremented by one after the access is complete.

8.2.7 I/O PORT ADDRESSING

The 3870 can use one of two forms of the I/O Port Address mode to transfer data between the Accumulator and an I/O port location. Instructions using this type of addressing may be two bytes long with the port address specified in the 8-bit operand field following the OP code, as shown below:



Alternatively, the instruction may use the short form to specify one of the first 16 I/O port locations with a single byte:



8.3 MK3870 INSTRUCTION TYPES

The 3870 instruction set consists of 76 different types of instructions. They can be classified into seven separate groups by function:

1) Arithmetic and Logical Group
2) Branch and Jump Group
3) Address Register Group
4) Accumulator Data Movement Group
5) Input/Output Group
6) CPU Control Group

The operation of the executable instructions in the 3870 is summarized in Table 8-3. An overview of the instruction set operation by group accompanies the summary. Notation used in this table is described in "Notes" at the end of this section.

8.3.1 ARITHMETIC AND LOGICAL GROUP

The Arithmetic and Logical Group of instructions allow the 3870 to perform various operations on data contained in the Accumulator and/or data from one of four sources within the device as listed below:

1) A Scratchpad Register
2) A location in Main Memory
3) An immediate value
4) The Data Counter Register

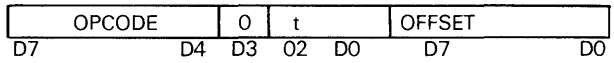
Data in a Scratchpad location may be added, added with decimal adjust, logically AND'ed, or Exclusive OR'ed with data contained in the Accumulator. Additionally, a scratchpad register may be decremented. Instructions in the Arithmetic and Logical group which operate on the Scratchpad utilize Scratchpad Addressing. The contents of the Accumulator may be added, added with decimal adjust, AND'ed, compared, OR'ed or exclusive OR'ed with a location in Main Memory. A location in Main Memory is accessed indirectly through the use of the Data Counter (Indirect Memory Addressing). The contents of the Accumulator may be complemented, incremented, shifted left or right, or AND'ed, OR'ed, Exclusive ORed, or compared with an immediate value. These types of instructions use either Implied or Immediate addressing. Finally, the contents of the Data Counter Register may be added with the contents of the Accumulator, which is treated as a signed binary (two's complement) number, with the result in the Data Counter. This instruction also uses Implied addressing.

8.3.2 BRANCH, JUMP, CALL, AND RETURN GROUP

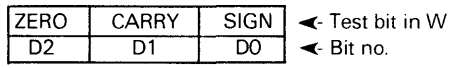
The Branch, Jump, Call, and Return Group include all of those instructions which can be used to transfer program control in the 3870.

Branch instructions which are conditional on the state of one or more of the bits in the Status register (W) are actually one of two basic instructions: Branch on True and Branch on False. The Branch on True instruction is a two byte instruction, with the branch condition contained in a field in the first byte of the instruction along with the OP code, and the second byte containing the relative offset pointing to the branch destination. The form of the Branch

on True instruction is shown below:



The three bit field "t" is actually a mask field for three of the bits which are to be tested in the Status Register. This three bit field is shown in detail below:



When a bit in the "t" field is set to a "1", the corresponding bit in the Status Register is tested during the execution of the Branch on True instruction. When a bit in the "t" field is set to a "0", the state of the corresponding bit in the Status Register is ignored. The operation of the Branch on True instruction is such that the branch to the branch destination address is taken if any of the unmasked bits in the Status Register are true. There are a total of 8 combinations of conditions which can be specified with the Branch on True instruction. Four of these conditions are often used in programming and given unique mnemonic names which are recognized by all 3870 cross assembler programs. These four forms of the Branch on True instruction are Branch on Carry (BC), Branch on Positive (BP) and Branch on Zero (BC). All of the possible branch conditions which exist for the Branch if True instructions are summarized in Table 8-1.

BRANCH CONDITIONS FOR BT INSTRUCTION

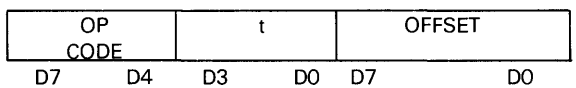
Table 8-1

OPERAND	STATUS FLAGS TESTED				
t	ZERO	CARRY	SIGN	DEFINITION	COMMENTS
0	0	0	0	Non-Functional	An effective 3 cycle NOP
1	0	0	1	Branch if Positive	Same as BP
2	0	1	0	Branch on Carry	Same as BC
3	0	1	1	Branch if Positive or on Carry	
4	1	0	0	Branch if Zero	Same as BZ
5	1	0	1	Branch if Positive	Same as t=1
6	1	1	0	Branch if Zero or on Carry	
7	1	1	1	Branch if Positive or on Carry	Same as t=3

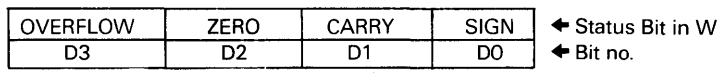
BRANCH CONDITIONS FOR BF INSTRUCTION
Table 8-2

OPERAND	STATUS FLAGS TESTED					
t	OVF	ZERO	CARRY	SIGN	DEFINITION	COMMENTS
0	0	0	0	0	Unconditional Branch Relative	
1	0	0	0	1	Branch on Negative	Same as BM
2	0	0	1	0	Branch if no Carry	Same as BNC
3	0	0	1	1	Branch if no Carry and Negative	
4	0	1	0	0	Branch if not Zero	Same as BZ
5	0	1	0	1		Same as t=1
6	0	1	1	0	Branch if no Carry and result is no Zero	
7	0	1	1	1		Same as t=3
8	1	0	0	0	Branch if there is no Overflow	Same as BNO
9	1	0	0	1	Branch if Negative and no Overflow	
A	1	0	1	0	Branch if no Overflow and no Carry	
B	1	0	1	1	Branch if no Overflow, no Carry and Negative	
C	1	1	0	0	Branch if no Overflow and not Zero	
D	1	1	0	1		Same as t=9
E	1	1	1	1	Branch if no Overflow, no Carry, and not Zero	
F	1	1	1	1		Same as t=B

The Branch if IS not = 7 instruction is used indicate whether or not Similarly, the Branch on False instruction consists of two bytes: The first byte specifies the OP code along with a four bit mask field for the OVERFLOW, ZERO, CARRY, AND SIGN bits in the Status Register. The form of the Branch on False instruction is illustrated below:



The mask field for the Branch on False instruction is:



The mask word selectively enables or disables the test on the bits in the Status Register as in the case of the Branch on True instruction. The operation of the Branch on False instruction is such that the branch to the destination address is taken if all of the unmasked bits in the Status Register are false (logic 0). Special forms of the Branch on False instruction which are given special mnemonic names include the Branch on Minus (BM), Branch on No Carry (BNC), Branch on Not Zero (BNZ), Branch if No Overflow (BNO), and Branch Relative (BR) instructions. All of the possible branch conditions which exist for the Branch if False instructions are summarized in Table 8-2. The branch condition for the BR7 instruction is true when ISC $\neq 7$. The branch is not not taken when ISC = 7, or when the lower half of the IS register is pointing to the end byte of an 8 byte block in the Scratchpad Register array. It is useful in many program sequences since only the lower half of the IS register can be auto-incremented or auto-decremented.

The Jump instruction allows program control to be transferred to any location within the 3870's internal Main Memory map. This is accomplished through the use of Extended Addressing. The user should take note of the fact that the contents of the Accumulator are modified during the execution of the Jump instruction. The Accumulator is used as a holding register for the most significant 8 bits of the destination address specified in the two byte operand field following the Jump OP code. Thus, this value results in the Accumulator when the Jump instruction is completed. The Jump Indirect (LR P0,Q) instruction can be used to transfer program control to the address in Main Memory which is pointed to by the contents of the Q Linkage Register.

Subroutines may be called either directly using the Call to Subroutine (PI) instruction or indirectly using the Call to Subroutine Indirect (PK) instruction. In either case the return address is placed in the Program Counter Stack Register during the execution of the instruction. The P register facilitates one level of subroutine calls or interrupts. Additional levels of subroutine and/or interrupts are possible by utilizing those instructions which transfer data between the P0 and P registers and the Linkage registers in the Scratchpad Array. These instructions are discussed in the Address Register Instruction description. For a detailed description of multiple level subroutine and interrupt implementation in the 3870, the user should refer to the Mostek application note "Multi-Level Subroutine and Interrupt Handling in the 3870". The user should note that the PI instruction modifies the contents of the Accumulator during the course of its execution. Like the Jump instruction, the Accumulator is used the hold the most significant portion of the destination address, or that portion of the address which is greater than the least significant 8 bits. Thus, this is the value which results in the Accumulator on completion of the instruction.

8.3.3 ACCUMULATOR DATA MOVEMENT GROUP

Instructions in the Accumulator Data Movement Group allow data to be moved between the Accumulator and either a scratchpad register or a location in Main Memory. An immediate 8 bit value may be moved into the Accumulator from the location in Main Memory following the instruction OP code. Data may be transferred between the Accumulator and any location in Main Memory by using the Load Memory (LM) and Store Memory (ST) instructions. The LM and ST instruction utilize Indirect Memory Addressing by pointing to the source/destination memory location with the Data Counter Register.

Data may also be transferred between the Accumulator and any Scratchpad location with instructions in the Accumulator Data Movement Group which use Scratchpad Addressing.

8.3.4 ADDRESS REGISTER GROUP

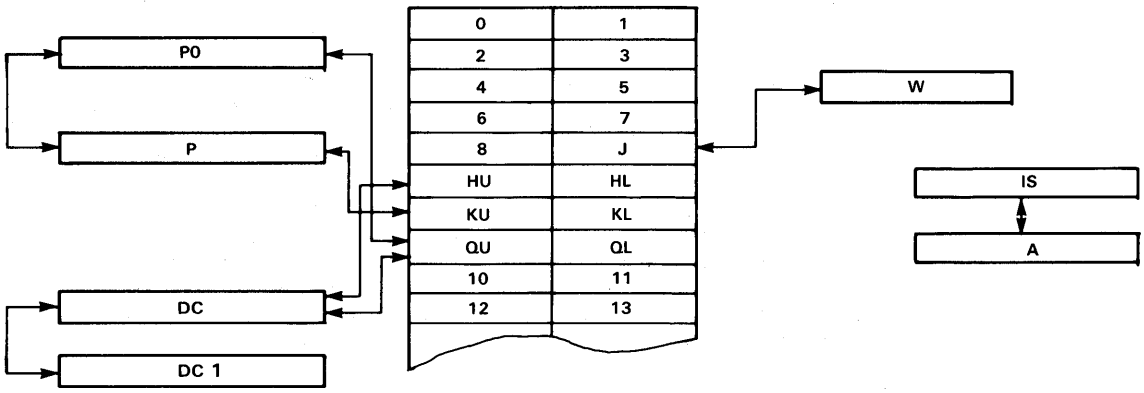
Intructions in the Address Register Group are used to manipulate the contents of registers within the 3870 which are used to address either the Scratchpad registers or Main Memory. The Indirect Scratchpad Address Register (IS) is used to address a location in the Scratchpad. The Program Counter Stack Register (P) is associated with the Program Counter and is primarily used for saving the return address during subroutine calls. The Data Counter (DC) and Auxiliary Data Counter (DC1) are used in address data constants in

Main Memory.

The Load IS Upper (LISU) and Load IS Lower (LISL) instructions can be used to selectively load either the upper three bits or the lower three bits of the IS Register. These instructions cause an immediate three bit value which is contained in the same byte as the OP code to be transferred into the appropriate half of the IS register. The contents of the IS register may be transferred to the Accumutor, or the contents of the Accumulator may be transferred into the IS register, by using the LR A,IS or LR IS,A instructions, respectively.

Data may be transferred between the P, DC, and DC1 registers and special register pairs which reside in the Scratchpad Register Array and are known as Linkage Registers. The Linkage registers are given special mnemonic names: H designates the register pair at locations 10 and 11, K designates the register pair at locations 12 and 13, and Q designates the register pair at locations 14 and 1. Figure 8-2 summarizes the data transfers which are possible between the Linkage Registers and the Address registers. For every such transfer shown in the diagram, there exists a corresponding instruction in the Address Register Group which facilitates that transfer, except for the LR PO,Q instruction, which is discussed with the Branch, Jump, Call, and Return instructions.

3870 ADDRESS REGISTER LINKAGES
Figure 8-2



8.3.5 INPUT/OUTPUT GROUP

There are four types of instructions in the Input/Output group which can access any I/O port location within the 3870. They are the Input (IN), Output (OUT), Input Short (INS), and Output Short (OUTS) instructions. The short form instructions can access any of the lower sixteen I/O port locations in the 3870 I/O port map and require only one byte. The long form instructions require two bytes; one for the OP code and the other to specify a port address from 0-255. In all existing 3870 devices, the short form instructions are sufficient to access all I/O port locations.

8.3.6 CPU CONTROL GROUP

The Enable Interrupts and Disable Interrupts instructions are included in the CPU Control Group. Also included are the instructions which transfer data between the Status Register W and Linkage Register J. These are the LR W,J and LR J,W instructions, which are primarily used to save and restore the status of the 3870 during interrupt subroutines. A No Operation instruction is included in the CPU Control Group.

MK3870 INSTRUCTION SET SUMMARY
Table 8-3

ARITHMETIC AND LOGICAL GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Add Carry	LNK	IMP	Add the Carry bit to the contents of the Accumulator
Add Immediate	AI ii	IMM	Add 8-bit immediate operand to the
Add to Data Counter	ADC	IND	Add the contents of the Accumulator to the Data Counter DC; result is in the Data Counter the contents of the Accumulator
Add Memory	AM	IND	Add the contents of memory location [DC] to the contents of the Accumulator
Add Memory Decimal	AMD	IND	Add the contents of memory location [DC] to the Accumulator with decimal adjust
Add Scratchpad	AS r	SCR	The contents of Scratchpad Register 'r' are added to the Accumulator
Add Scratchpad Decimal	ASD r	SCR	The contents of Scratchpad Register 'r' are added to the Accumulator with decimal adjust
And Immediate	NI ii	IMM	Perform the logical AND of the 8-bit immediate operand and the contents of the Accumulator
And Memory	NM	IND	Perform the logical AND between memory location [DC] and the Accumulator
And Scratchpad	NS r	SCR	Scratchpad location 'r' is logically ANDed with the contents of the Accumulator
Compare Immediate	CI ii	IMM	Non-destructive subtraction of the Accumulator from the immediate operand
Compare Memory	CM	INC	Non-destructive subtraction of the Accumulator from the contents of memory location [DC]
Complement	COM	IMP	Performs a one's complement operation on the contents of the Accumulator
Decrement Scratchpad	DS r	SCR	The contents of Scratchpad Register are decremented by 1
Exclusive OR Immediate	XI ii	IMM	Performs a logical Exclusive OR operation of the 8 bit immediate operand and the Accumulator
Exclusive OR Memory	XM	IND	Perform a logical Exclusive OR operation between the Accumulator and memory location [DC]

ARITHMETIC AND LOGICAL GROUP (Continued)

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Exclusive OR Scratchpad	XS r	SCR	Scratchpad location 'r' is logically exclusive OR'ed with the contents of the Accumulator
Increment	INC	IMP	Add the value of 1 to the Accumulator
OR Immediate	OI ii	IMM	Perform a logical OR of the 8-bit immediate operand and the Accumulator
OR Memory	OM	IND	Perform a logical OR operation between the Accumulator and memory location [DC]
Shift Left 1	SL 1	IMP	Shift the contents of the Accumulator left by one
Shift Left 4	SL 4	IMP	Shift the contents of the Accumulator left by four
Shift Right 1	SR 1	IMP	Shift the contents of the Accumulator right by one
Shift Right 4	SR 4	IMP	Shift the contents of the Accumulator right by four

BRANCH, JUMP, CALL, AND RETURN GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Branch Relative	BR aa	REL	Branch unconditionally
Branch if False	BF taa	REL	Branch if all of the unmasked Status bits = 0
Branch if Minus	BM aa	REL	Branch if the Sign bit = 0
Branch if No Carry	BNC aa	REL	Branch if the Carry bit = 0
Branch if No Overflow	BNO aa	REL	Branch if the Overflow bit = 0
Branch if Not Zero	BNZ aa	REL	Branch if the Zero bit = 0
Branch if ISL not = 7	BR7 aa	REL	Branch if the value of the lower half of the IS register is not = 7
Branch on Carry	BC aa	REL	Branch if Carry bit = 1
Branch on Positive	BP aa	REL	Branch if the Sign bit = 1
Branch on True	BT taa	REL	Branch if any unmasked Status bit = 1
Branch on Zero	BZ aa	REL	Branch if the Zero bit = 1
Jump	JMP aaaa	EXT	Program Counter is loaded with the immediate value 'aaaa', the Accumulator is loaded with upper half of aaaa

BRANCH, JUMP, CALL, AND RETURN GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Jump Indirect	LR P0,Q	IMP	The Program Counter is loaded with the contents of Linkage Register Q
Call to Subroutine	PI iiii	EXT	The Program Counter is loaded with the value 'iiii'; the return address is saved in P; A is destroyed
Call to Subroutine Indirect	PK	IND	The Program Counter is loaded with the contents of Linkage Register K; the return address is saved in P
Return from Subroutine	POP	IMP	Swap the contents of the Program Counter with the Stack Register P

ACCUMULATOR DATA MOVEMENT GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Clear	CLR	IMM	Load Accumulator immediate with zero
Load	LR A,KU	IMP	The Accumulator is loaded with the contents of the upper half of Linkage Register K
Load	LR A,KL	IMP	The Accumulator is loaded with the contents of the lower half of Linkage Register K
Load	LR KU,A	IMP	The upper half of Linkage register K is loaded with the contents of the Accumulator
Load	LR KL,A	IMP	The lower half of Linkage register K is loaded with the contents of the Accumulator
Load	LR A,QU	IMP	The Accumulator is loaded with the contents of the upper half of Linkage Register Q
Load	LR A,QL	IMP	The Accumulator is loaded with the contents of the lower half of Linkage Register Q
Load	LR QU,A	IMP	The upper half of Linkage register Q is loaded with the contents of the Accumulator
Load	LR QL,A	IMP	The upper half of Linkage register Q is loaded with the contents of the Accumulator
Load	LR A,r	SCR	The Accumulator is loaded with the contents of Scratchpad Register 'r'
Load	LR r,A	SCR	Scratchpad Register 'r' is loaded with the contents of the Accumulator
Load Immediate	LI ii	IMM	Load the value of the 8-bit immediate operand to the Accumulator

ACCUMULATOR DATA MOVEMENT GROUP (Continued)

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Load Immediate Short	LIS 0i	IMM	Load the value of the 8-bit immediate operand to the Accumulator
Load Memory	LM	IND	Load Accumulator with the contents of memory location [DC]
Store Memory	ST	IND	Store the value of the Accumulator in memory location [DC]

ADDRESS REGISTER GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Load Data Counter Immediate	DCI iiii	IMM	The Data Counter is loaded with the immediate value 'iiii'
Exchange DC	XDC	IMP	Swap the contents of DC with DC1
Load Data Counter	LR DC,Q	IMP	The Data Counter is loaded with the contents of linkage register Q
Store Data Counter	LR Q,DC	IMP	Linkage Register Q is loaded with the contents of the Data Counter
Load Data Counter	LR DC,H	IMP	The Data Counter is loaded with the contents of Linkage Register H
Store Data Counter	LR H,DC	IMP	Linkage Register H is loaded with the contents of the Data Counter
Load IS Lower	LISL bbb	IMM	Load the lower half of the IS register with the immediate value 'bbb'
Load IS Upper	LISU bbb	IMM	Load the upper half of the IS register with the immediate value 'bbb'
Load IS	LR IS,A	IMP	Load the IS register with the contents of the Accumulator
Store IS	LR A,IS	IMP	The Accumulator is loaded with the contents of the IS register
Load Stack Register	LR P,K	IMP	The Stack Register P is loaded with the contents of Linkage Register K
Store Stack Register	LR K,P	IMP	Linkage Register K is loaded with the contents of the Stack Register

INPUT/OUTPUT GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Input	IN aa	IOP	The contents of Port 'aa' are loaded (04-FF) into the Accumulator
Input Short	INS a (0-F)	IOP	The contents of Port 'a' are loaded into the Accumulator
Output	OUT aa (04-FF)	IOP	The contents of the Accumulator are output to Port 'aa'
Output Short	OUTS a (00-04)	IOP	The contents of the Accumulator are output to Port 'a'

CPU CONTROL GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Disable Interrupts	DI	IMP	Interrupts to the 3870 are disabled; ICB is reset to 0
Enable Interrupts	EI	IMP	Interrupts are enabled to the 3870; ICB is set to 1
Load Status Register	LR W,J	IMP	The Status Register is loaded with the contents of Linkage Register J
No Operation	NOP	IMP	The contents of the Program Counter are incremented
Store Status Register	LR J,W	IMP	Linkage Register J is loaded with the contents of the Status Register

8.4 INSTRUCTION EXECUTION AND TIMING

A detailed summary of the timing and execution of the 3870 instruction set is included in Table 8-4. This table provides OP codes, instruction cycle sequence, effect on Status flags, and operation information for each and every 3870 instruction. The information contained in each of the columns contained in the table is described below;

OP CODE: The mnemonic name for the instruction is contained in this column.

OPER: The symbolic name for the operand(s) used in the instruction appear in this column.

OBJECT CODE: The hexadecimal equivalent of the machine code to which the instruction translates.

CYCLE: The sequence of machine cycles which occur during the course of execution of the instruction. A Long cycle is symbolized with an "L" and a Short cycle is symbolized with an "S". The order in which these cycles occur during instruction execution appear downward in this column. e.g: The instruction LM is executed with a Long cycle (L), then a Short cycle (S).

μS: The execution time of the instruction is indicated in this column in units of microseconds. This execution time is based on using a time base frequency of 4 MHz, yielding an internal Φ clock frequency of 2 MHz. Execution time which results from a different time base frequency may be calculated by multiplying this time by the value of (time base freq.)/4 MHz.

STATUS FLAGS: The effect which the instruction has on the four flags in the Status register is shown in this column. (See notes for terminology explanation.)

INT: If the instruction is privileged, it is denoted with the letter "p" in this column.

FUNCTION: The operation of the instruction is described symbollicaly in this column. (See Notes for explanation of terminology).

INSTRUCTION TIMING AND EXECUTION
Table 8-4

ARITHMETIC AND LOGICAL GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
ADC		8E	L S	5	-	-	-	-		C ← DC + A
AI	ii	24	L S	5	↕	↕	↕	↕		A ← A + ii
AM		88	L S	5	↕	↕	↕	↕		A ← A + [DC]
AMD		89	L S	5	↕	↕	↕	↕		A ← A + [DC] w/BCD adjust
AS	r	Cr	S	2	↕	↕	↕	↕		A ← A + r
ASD	r	Dr	S S	4	↕	↕	↕	↕		A ← A + r w/BCD adjust
CI	ii	25ii	L S	5	↕	↕	↕	↕		ii - A

ARITHMETIC AND LOGICAL GROUP (Continued)

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
CM		8D	L S	5	↕	↕	↕	↕		[DC] - A
COM		18	S	2	0		0			A ← A*
DS	r	3r	L	3	↕	↕	↕	↕		r ← r - 1
INC		1F	S	2	↕	↕	↕	↕		A ← A + 1
LNK		19	S	2	↕	↕	↕	↕		A ← A + CRY
NI	ii	21ii	L S	5	0	↕	0	↕		A ← A © ii
NM		8A	L S	5	0	↕	0	↕		A ← A © [DC]
NS	r	Fr	S	2	0	↕	0	↕		A ← A © r
OI	ii	22ii	L S	5	0	↕	0	↕		A ← A v ii
OM		8B	L S	5	0	↕	0	↕		A ← A v [DC]
SL	1	13	S	2	0	↕	0	↕		Shift 'A' left by 1 LSB ← 0
SL	4	15	S	2	0	↕	0	↕		Shift 'A' left by 4 LS nibble ← 0000
SR	1	12	S	2	0	↕	0	1		Shift 'A' right by 1 MSB ← 0
SR	4	14	S	2	0	↕	0	1		Shift 'A' right by 4 MS nibble ← 0000
XI	ii	23ii	L S	5	0	↕	0	↕		A ← A (+) ii
XM		8C	L S	5	0	↕	0	↕		A ← A (+) [DC]
XS	r	Er	S	2	0	↕	0	↕		A ← A (+) r

BRANCH, JUMP, CALL, AND RETURN GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
BC	aa	82aa	S S/L S	6/7	-	-	-	-		Branch if C = 1
BF	aa	9taa	S S/L S	6/7	-	-	-	-		Branch if unmasked status it is false
BM	aa	91aa	S S/L S	6/7	-	-	-	-		Branch if S = 0
BNC	aa	92aa	S S/L S	6/7	-	-	-	-		Branch if C = 0
BNO	aa	98aa	S S/L S	6/7	-	-	-	-		Branch if O = 0
BNZ	aa	94aa	S S/L S	6/7	-	-	-	-		Branch if Z = 0
BP	aa	81aa	S S/L S	6/7	-	-	-	-		Branch if S = 1
BR	aa	90aa	S L	7	-	-	-	-		Branch always

BRANCH, JUMP, CALL, AND RETURN GROUP (Continued)

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
			S							
			S							
BT	aa	8taa	S	6/7	-	-	-	-		Branch if any
			S/L							unmasked status it
			S							is true
BZ	aa	84aa	S	6/7	-	-	-	-		Branch if Z = 1
			S/L							
			S							
JMP	aaaa	29aaaa	L	11	-	-	-	-	p	P0 ← aaaa;
			L							A ← P0 upper
			L							
			S							
LR	P0,Q	0D	L	8	-	-	-	-		P0 ← Q
			L							
			S							
PI	iiii	28iiii	L	13	-	-	-	-	p	P ← P0;
			S							P0 ← iiii
			L							
			L							
			S							
PK		0C	L	8	-	-	-	-	p	P ← P0;
			L							P0 ← K
			S							
POP		1C	S	4	-	-	-	-	p	P0 <=> P
			S							

NOTE:

In all conditional branch instructions, two possible execution times are given. The shorter time corresponds to the execution time which results when the branch is not taken. This corresponds to the fact that a short cycle is executed in this case. When the branch is taken, a long cycle is executed, and the execution time is longer.

ACCUMULATOR DATA MOVEMENT GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
CLR	70	70	S	2	-	-	-	-		A ← 00
LR	A,KU	00	S	2	-	-	-	-		A ← KU
LR	A,KL	01	S	2	-	-	-	-		A ← KL
LR	KU,A	04	S	2	-	-	-	-		KU ← A
LR	KL,A	05	S	2	-	-	-	-		KL ← A
LR	A,QU	02	S	2	-	-	-	-		A ← QU
LR	A,QL	03	S	2	-	-	-	-		A ← QL
LR	QU,A	06	S	2	-	-	-	-		QU ← A
LR	QL,A	07	S	2	-	-	-	-		QL ← A
LR	A,r	4r	S	2	-	-	-	-		A ← r

ACCUMULATOR DATA MOVEMENT GROUP (Continued)

LR	r,A	5r	S	2	-	-	-	-		r ← A
LI	ii	20ii	L S	5	-	-	-	-		A ← ii
LIS	i	7i	S	2	-	-	-	-		A ← 7i
LM		16	L S	5	-	-	-	-		A ← [DC]
ST		17	L S	5	-	-	-	-		[DC] ← A

ADDRESS REGISTER GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
DCI	iiii	2Aiiii	L S L S S	12	-	-	-	-		DC ← iiii
XDC		2C	S S	4	-	-	-	-		DC <=> DC1
LR	DC,Q	0F	L L S	8	-	-	-	-		DC ← Q
LR	Q,DC	0E	L L S	8	-	-	-	-		Q ← DC
LR	DC,H	10	L L S	8	-	-	-	-		DC ← H
LR	H,DC	11	L L S	8	-	-	-	-		H ← DC
LISL		6(0bbb)	S	2	-	-	-	-		ISL ← bbb
LISU		6(1bbb)	S	2	-	-	-	-		ISU ← bbb
LR	A,IS	0A	S	2	-	-	-	-		A ← IS
LR	IS,A	0B	S	2	-	-	-	-		IS ← A
LR	K,P	0 8	L L S	8	-	-	-	-		K ← P
LR	P,K	09	L L S	8	-	-	-	-		P ← K

INPUT/OUTPUT GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
IN	pp	26pp	L L S	8	0		0			A ← pp
INS	0,1	A0,A1	S S	4	0		0			A ← [0,1]
INS	p	Ap	L L S	8	0		0			A ← [p] p > 1

INPUT/OUTPUT GROUP (Continued)

OUT	pp	27pp	L L S	8	-	-	-	-	p	pp ← A
OUTS	0,1	B0,B1	S S	4	-	-	-	-	p	p0,p1 ← A
OUTS	2-15	B2-BF	L L S	8	-	-	-	-	p	p0-pF ← A

CPU CONTROL GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
DI		1A	S	2	-	-	-	-		Clear ICB
EI		1B	S	2	-	-	-	-	p	Set ICB
LR	J,W	1E	S	2	-	-	-	-		J ← W
LR	W,J	1D	S	4	-	-	-	-	p	W ← J
NOP		2B	S	2	-	-	-	-		P0 ← P0 + 1

NOTES

p Denotes privileges instruction

Status legend:

m = test and modify according to result
? = unknown
- = not altered
0 = reset to 0
1 = set to 1

Function symbology

←	is loaded with
↔	is exchanged with
[ ]	the contents of the location pointed to by
©	logical AND
v	logical OR
a	
a	Address variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Memory Address Register)
DC1	Auxiliary Data Counter
H	Scratchpad register pair 10 and 11 (Linkage Register)
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least significant 3 bits of IS
ISU	Most significant 3 bits of IS
J	Scratchpad Register 9
K	Scratchpad Register pair 12 and 13 (Linkage Register)
KL	Scratchpad Register 13 (K Lower)
KU	Scratchpad Register 12 (K Upper)
P0	Program Counter
P	Program Counter Stack Register
p I/O	Port location
Q	Scratchpad Register pair 14 and 15 (Linkage Register)

QL	Scratchpad Register 15 (Q Lower)
QU	Scratchpad Register 14 (Q Upper)
r	Scratchpad Register (any address 0 thru b; see below)
W	Status Register

3870 Addressing Modes (Abbreviations)

IMM	Immediate Addressing
IMP	Implied Addressing
REL	Relative Addressing
EXT	Extended Addressing
SCR	Scratchpad Addressing (see below)
INM	Indirect Memory Addressing
IOP	I/O Port Addressing

Scratchpad Addressing Using IS (r not = 0 thru B)

r = C (Hex)	Register Addressed by IS (IS is unmodified)
r = D	Register Addressed by IS (IS is incremented)
r = E	Register Addressed by IS (IS is decremented)
r = F	Register Legal OP Code