

Chapter 20

THE HEWLETT PACKARD MC2

The MC2 is the first microprocessor manufactured by Hewlett Packard. The MC2 is a 16-bit microprocessor designed specifically for process control applications; it has been designed for internal use within the many instruments and electronic products manufactured by Hewlett Packard. The MC2 is most unlikely to be sold as a single chip in the foreseeable future; even its availability as a computer card is not guaranteed at the present time.

The most important aspect of the MC2 is its technology; it is built using CMOS logic with Silicon On Sapphire (SOS technology). The CMOS logic gives the MC2 typical CMOS low power requirements and noise insensitivity, while Silicon On Sapphire technology gives it high speed.

Using a +12V power supply the MC2 operates with a 125 nanosecond clock and executes instructions in 4 to 12 clock cycles. Typical instructions are therefore executed in less than one microsecond.

The MC2 is packaged on a squared, 48-pin leadless ceramic substrate.

The sole manufacturer of the MC2 is:

HEWLETT PACKARD COMPANY
Data Systems Division
11000 Wolfe Road
Cupertino, CA 95014

A second source for this microprocessor is unlikely in the foreseeable future.

AN MC2 SYSTEM OVERVIEW

Logic implemented on the MC2 CPU chip is illustrated in Figure 20-1. A number of support devices for the MC2 have been manufactured, but no information on these support devices is available at the present time.

Clock logic is external to the MC2 chip; however a simple single waveform external clock signal will suffice.

Figure 20-1 shows I/O interface logic as being implemented on the MC2 CPU chip. This reflects the unusual way in which the MC2 handles external devices. The MC2 CPU has eight 16-bit general-purpose programmable registers. Every I/O device is assumed to have a CPU equivalent set of eight programmable registers. Instructions and control signals of the MC2 treat registers of the CPU and I/O devices similarly, which means that no special I/O interface logic is required.

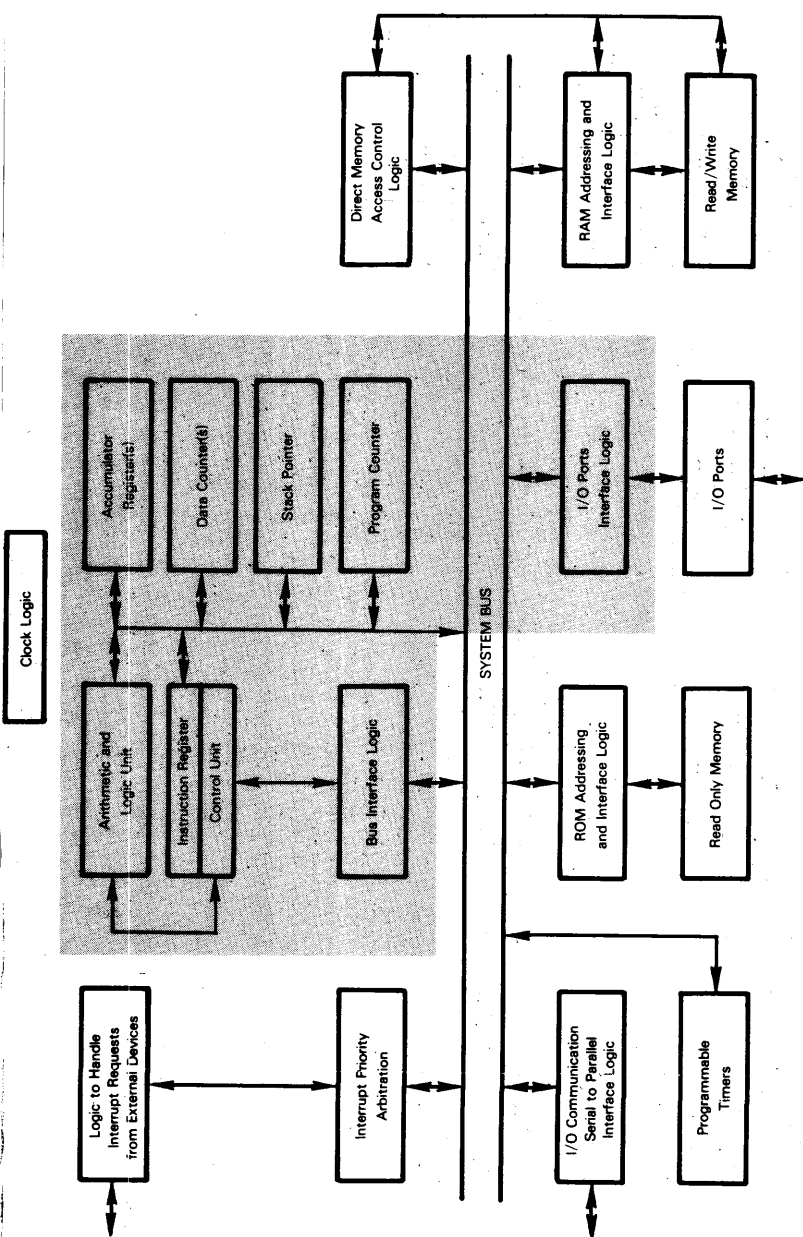
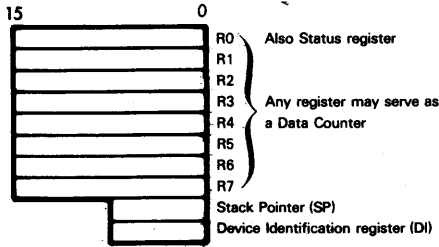


Figure 20-1. Logic Of The Hewlett Packard MC2 Microprocessor

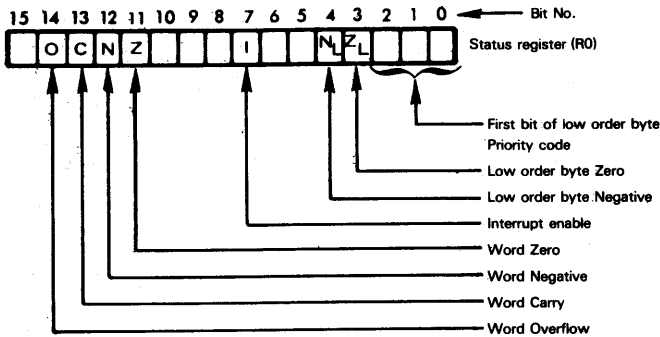
MC2 PROGRAMMABLE REGISTERS AND STATUS

The MC2 has eight 16-bit programmable registers and an 8-bit Stack Pointer. In addition there is an 8-bit I/O Device Identification register. Registers may be illustrated as follows:



Any one of the eight 16-bit registers may be used as a Data Counter.

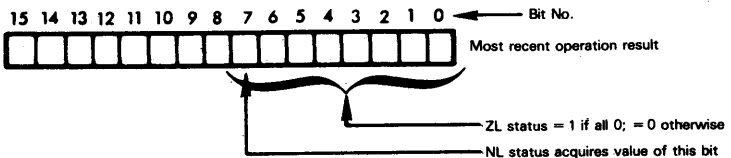
Register R0 serves as the Status register. Its contents are interpreted as follows:



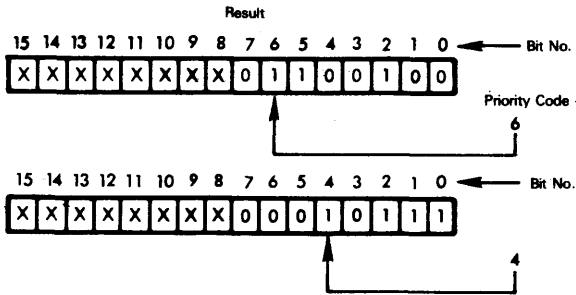
The Zero, Negative, Carry and Overflow statuses in bits 11, 12, 13 and 14, respectively, apply to the 16-bit result of the most recent data operation performed. Zero, Carry and Overflow are standard statuses, as described in Volume 1, Chapter 6. The Negative status reflects the sign of the 16-bit result; that is to say, it is set to the value of the result's high order bit.

The Interrupt Enable flag in Status register bit 7 is set to 1 in order to enable interrupts. It is reset to 0 in order to disable interrupts.

The low order byte Zero and Negative statuses are identical to standard Zero and Negative statuses, except that they reflect the low order 8 bits of the most recent operation's result. This may be illustrated as follows:



The low order three Status register bits are referred to as a Priority Code. This Priority Code identifies the highest order 1 bit in the low order byte of the most recent operation's result. The Priority Code has the value of the bit position being identified. Here are some examples of Priority Codes:



The Stack Pointer enables a 256-word Stack. The Stack occupies the first 256 words of read/write memory, with memory word 0 being the top of the Stack.

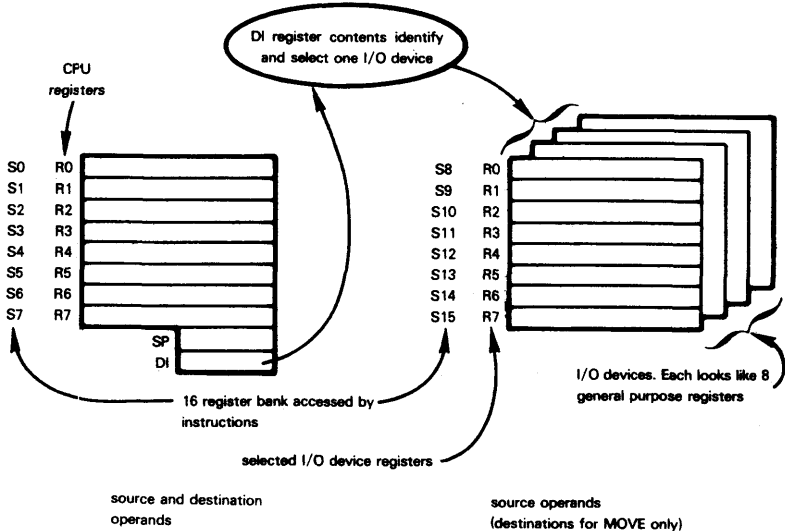


Figure 20-2. CPU And I/O Device Registers' Organization For The MC2

The I/O Device Identification register, also referred to as a Base register, identifies one of 256 possible external I/O devices. The identified external I/O device will be interpreted as consisting of eight 16-bit registers. When executing Register-Register instructions, there is little differentiation made between the eight CPU registers and the eight registers of the identified external device. The two sets of eight registers constitute a 16-register bank out of which two operands are selected. The two operands may or may not come from the same register. The destination, which is the first identified operand register, is usually one of the CPU registers (R0 - R7); only the Register-Register Move instruction permits an external register (R8 - R15) to be the destination. This scheme is illustrated in Figure 20-2.

MC2 MEMORY ADDRESSING MODES

The MC2 is quite limited in its memory reference capabilities. Instructions allow you to load data from memory into a CPU register, or to store data from CPU registers to memory. **Data access instructions use direct memory addressing or implied memory addressing.**

Direct memory addressing instructions are two words long; the second instruction object code word provides the 16-bit direct memory address.

Instructions that use implied memory addressing allow any one of the eight CPU registers to specify the 16-bit memory address.

Conditional Branch instructions and Subroutine Call instructions allow direct and indirect addressing; however direct addressing is program relative and the displacement is an 8-bit signed binary number.

HARDWARE ASPECTS OF THE MC2

We are not going to describe pins and signals of the MC2 because Hewlett Packard has not made sufficient information available at the present time. Also, such information will be irrelevant until you can buy the MC2 as a chip. **Instead we will provide a brief summary of principal MC2 hardware characteristics.**

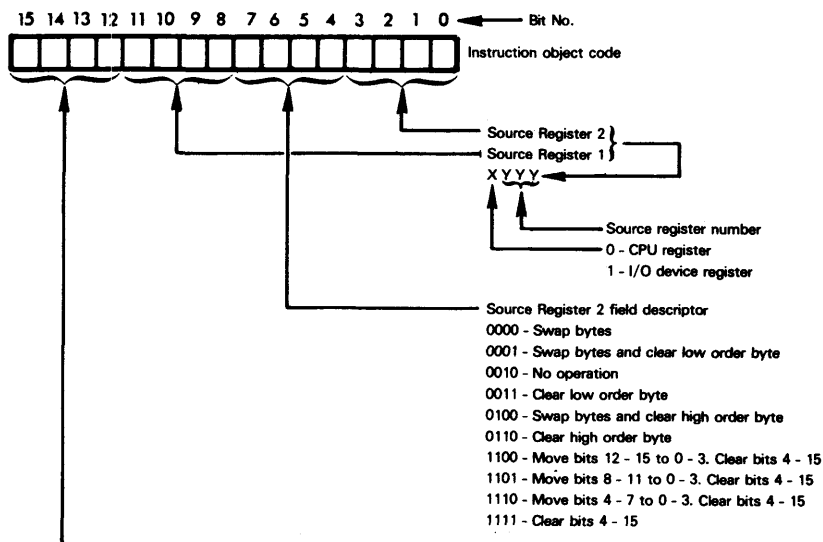
The MC2 is packaged as a 48-pin package. This allows separate 16-bit Data and Address Busses, together with an adequate set of control signals. Control logic on the System Bus is asynchronous, having a request/acknowledge control philosophy. This simplifies multiple CPU configurations. Execution of a "No Operation" instruction puts any CPU into a slave state on the System Bus, at which time its internal registers may be accessed by some other "master" CPU. A slave CPU may be converted into the master via an interrupt request.

MC2 interrupt logic is primitive but effective. There is one interrupt request line; when an interrupt is acknowledged, a subroutine call to memory location $FFFE_{16}$ is executed. Memory location $FFFE_{16}$ must contain the beginning address for the interrupt service routine.

THE MC2 INSTRUCTION SET

The MC2 instruction set is characterized by a lack of distinction between CPU registers and I/O devices. Most instructions that operate on data or move data are Register-Register instructions; as illustrated in Figure 20-2, each register may be an internal CPU register or a register out of an I/O device. Thus there is no difference between Move instructions that access two CPU registers, one CPU register and an I/O device or two registers from the same I/O device. This similarity between Register-Register and I/O instructions is manifest by the way in which the MC2 instruction set has been defined in Table 20-1.

For a better understanding of MC2 instructions you must understand the way in which Register-Register instruction object codes are defined. This may be illustrated as follows:



Source registers 1 and 2 may each be identified as any one of the eight CPU 16-bit registers, or any one of the eight I/O device 16-bit registers. The particular I/O device which is currently selected is defined by the contents of the I/O Device Identification register.

Source register 1 also becomes the Destination register for the result of the operation. In all instructions except MOVE, Source register 1 must be one of the eight CPU registers.

The contents of Source register 2 are manipulated before becoming an operand for the operation to be performed. The manipulation performed on Source register 2 contents is defined by the field descriptor. Register-Register instructions therefore perform an operation identified by bits 12 through 15 of the instruction object code; the operation uses two 16-bit operands as inputs. These two operands may come from the CPU registers, or the currently selected I/O device's registers. One 16-bit operand may be manipulated as defined by the field descriptor before it becomes an input to the operation specified by the instruction code.

THE BENCHMARK PROGRAM

For the Hewlett Packard MC2 our benchmark program may be illustrated as follows:

	LDWI	R1 = IOBUF	LOAD INPUT BUFFER ADDRESS IN REGISTER 1
	LOAD	R2 = COUNT	LOAD BUFFER SIZE IN REGISTER 2
	LOAD	R3 = TABLE	LOAD NEXT FREE TABLE LOCATION IN REGISTER 3
LOOP:	LOAD	R4 = (R1)	INPUT DATA WORD TO REGISTER 4
	STOR	(R3) = R4	STORE WORD TO NEXT FREE TABLE LOCATION
	ADDI	R1,1	INCREMENT BUFFER ADDRESS
	ADDI	R3,1	INCREMENT TABLE ADDRESS
	SUBI	R2,1	DECREMENT WORD COUNT
	CBR	LOOP IF G	RE-ITERATE IF COUNT STILL GREATER THAN ZERO
	STOR	TABLE = R3	SAVE ADDR OF NEXT TABLE WORD

This benchmark program makes very few assumptions. Memory is addressed in 16-bit units (rather than 8-bit bytes), and data is transferred 16 bits at a time. The input table IOBUF and the data table TABLE can have any length, and can reside anywhere in memory. The address of the first free word in TABLE is stored in the first word of the TABLE

The following notation is used in Table 20-1:

BYTE	8 bits of immediate data — the lower byte of the instruction word.
CDST CREG CSRC	Any of the CPU registers (Registers 0 through 7).
DST	Any of the 16 registers, used as the destination of a move or result.
DI	The I/O Device Identification register (Base register).
F	Fill specification for register shift: if F is 0, the bit is reset to 0; if F is 1, the bit is set to 1.
<.FD>	Optional field descriptor: SWB Swap bytes LJL Left justify lower byte LJU Left justify upper byte RJU Right justify upper byte RJL Right justify lower byte RJ0 Right justify high order nibble RJ1 Right justify next-to-high order nibble RJ2 Right justify next-to-low order nibble RJ3 Right justify low order nibble
REG (FD)	The result of the operation of the field descriptor on the specified register.
SRC (K)	Operand field specification of one bit of the register. Register may be any of the 16 registers; K may be any number from 0 to 15.
SRC <K>	Bit K of Register SRC.
<.I>	Optional indirection specification. When I is present, the address used is the contents of the memory location addressed.

<.IF CC>	Optional Condition Code representing a linear combination of the Zero and Negative status flags and "Greater Than":		
N	Z	<u>NVZ</u>	
0	0	0	Not true
0	0	1	G - greater than 0
0	1	0	E - equal to 0
0	1	1	GE - greater than or equal to 0
1	0	0	L - less than 0
1	0	1	LG - not equal to 0
1	1	0	LE - less than or equal to 0
1	1	1	Unconditional branch
LABEL	A 16-bit address; it may be the second word in the instruction, or its lower byte may be the lower byte of a one-word instruction.		
REG	Any of the CPU or external registers.		
<(REG<,FD>)>	Optional indexing specification. For example, the instruction: IBR TABLE(9,RJL) will calculate an address by clearing the upper byte of the contents of Register 9 and adding the result to the 16-bit word TABLE. Then the contents of the location thus addressed will be the address at which instruction execution continues.		
RO	Register 0, the Status register, as described earlier in this chapter.		
PC	The Program Counter.		
SP	The Stack Pointer.		
SRC	Any of the 16 registers, used as the source of an operand.		
STATUSES	The following status flags are affected by the instructions: O The Overflow status C The Carry status N The Negative or Sign status Z The Zero status L The lower byte statuses NL and ZL The following symbols are used in the STATUSES columns: A blank means the status flag is not affected by the operation. X The operation affects the status flag in a meaningful manner. ? The operation affects the status flag, but it is meaningless.		
WORD	16 bits of immediate data.		
x <y,z>	Bits y through z of the Register x. For example, PC<15,8> represents the upper byte of the Program Counter.		
[]	Contents of location enclosed within brackets. If a register designation is enclosed within the brackets, then the designated register's contents are specified. If a memory address is enclosed within the brackets, then the contents of the addressed memory location are specified.		
[[]]	Implied memory addressing; the contents of the memory location designated by the contents of a register.		
Λ	Logical AND		
V	Logical OR		
⊕	Logical Exclusive-OR		
←	Data is transferred in the direction of the arrow.		
**	Exponentiation. 2**K represents a 16-bit word with a 1 in bit K, and 0 in all the other bits.		

Table 20-1. A Summary Of The MC2 Instruction Set

TYPE	MNEMONIC	OPERAND(S)	BYTES	STATUSES					OPERATION PERFORMED
				O	C	N	Z	L	
PRIMARY MEMORY REFERENCE	LOAD	CDST = LABEL <(CSRC < FD >) >	4						[CDST] ← [LABEL + (CSRC(FD))] Load CPU register from memory using direct addressing or indexed addressing via a CPU register.
	LOAD	CDST = (CSRC)	2						[CDST] ← [(CSRC)] Load CPU register from memory using implied addressing via a CPU register.
	STOR	LABEL <(CDST < FD >) > = CSRC	4						[LABEL + (CDST(FD))] ← [CSRC] Store CPU register to memory using direct addressing or indexed addressing via a CPU register.
	STOR	(CDST) = CSRC	2						[(CDST)] ← [CSRC] Store CPU register to memory using implied addressing via a CPU register.
I/O AND IMMEDIATE	LDWI	CDST = WORD	4						[CDST] ← WORD Load immediate 16 bits to CPU register.
	LDBI	REG = BYTE	2	?	?	X	X	X	[REG < 7,0 >] ← BYTE Load immediate 8 bits to CPU or external register.
IMMEDIATE OPERATE	ADDI	CDST, BYTE	2	X	X	X	X	X	[CDST] ← [CDST] + BYTE Add immediate 8 bits to lower half of CPU register.
	SUBI	CDST, BYTE	2	X	X	X	X	X	[CDST] ← [CDST] - BYTE Subtract immediate 8 bits from lower half of CPU register.
	CMPI	CDST, BYTE	2	X	X	X	X	X	[CREG] - BYTE Compare immediate 8 bits with lower half of CPU register. Only the statuses are affected.
	BR	REG < FD >	2						[PC] ← [REG(FD)] Branch to memory location addressed by register contents or by some operation on the register's contents.
I/O, JUMP, AND BRANCH ON CONDITION	IBR	LABEL <(REG < FD >) >	4						[PC] ← [LABEL + (REG(FD))] Branch using direct addressing or indexed addressing via any CPU or external register.

Table 20-1. A Summary Of The MC2 Instruction Set (Continued)

TYPE	MNEMONIC	OPERAND(S)	BYTES	STATUSES								OPERATION PERFORMED
				O	C	N	Z	L				
I/O, JUMP, AND BRANCH ON CONDITION (CONTINUED)	CALL	LABEL <I> <IF CC>	2	?	?	?	?	?			If CC is true then [SP] ← [RO] [SP] ← [SP] + 1 [RO] ← [PC] [PC] ← [PC] <15,8> LABEL <7,0> or [PC] ← [[PC] <15,8> LABEL <7,0>] if I is specified Subroutine call — may be conditional or unconditional. If condition is not satisfied, Program Counter is incremented and next instruction is executed. If condition is satisfied, statuses are saved on the stack, the incremented Program Counter is saved in Register 0, and the lower 8 bits of the Program Counter are replaced by the second byte of the instruction. Subroutine starting location must be within 256 words of CALL instruction location. [PC] ← [CREG] [CREG] ← [SP] - 1 [SP] ← [SP] - 1 Subroutine return — get return address from specified CPU register and pop the stack into that register. If CC is true then [PC] ← [PC] <15,8> LABEL <7,0> or [PC] ← [[PC] <15,8> LABEL <7,0>] if I is specified Conditional branch — if condition is satisfied, then replace lower 8 bits of Program Counter with lower 8 bits of instruction. Branch destination must be within 256 words of CBR instruction.	
	RTN	CREG	2									
		CBR	LABEL <I> <IF CC>	2								
I/O AND REGISTER-REGISTER MOVE	MOVE	DST = SRC <FD>	2	?	?	X	X	X			[DST] ← [SRC/FD] Move data from register to register, optionally operating on source word.	
	STRB	CDST									[CDST] ← [DI] Move contents of Device Identification register into specified CPU register.	
	LDRB	CSRC <FD>	2								[DI] ← [CSRC/FD] Load Device Identification register with contents of a CPU register, or with some function of those contents.	

Table 20-1. A Summary Of The MC2 Instruction Set (Continued)



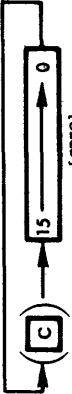
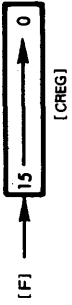
TYPE	MNEMONIC	OPERAND(S)	BYTES	STATUSES					OPERATION PERFORMED
				O	C	N	Z	L	
I/O AND REGISTER-REGISTER OPERATE	ADD	CDST.SRC <, FD >	2	X	X	X	X	X	$[CDST] \leftarrow [SRC(FD)] + [CDST]$ Add contents of CPU register and any register; deposit result in CPU register.
	SUBR	CDST.SRC <, FD >	2	X	X	X	X	X	$[CDST] \leftarrow [SRC(FD)] - [CDST]$ Subtract contents of CPU register from any register's contents; deposit result in CPU register.
	AND	CDST.SRC <, FD >	2	?	?	X	X	X	$[CDST] \leftarrow [SRC(FD)] \wedge [CDST]$ AND CPU register contents with any register's contents; deposit result in CPU register.
	OR	CDST.SRC <, FD >	2	?	?	X	X	X	$[CDST] \leftarrow [SRC(FD)] \vee [CDST]$ OR CPU register contents with any register's contents; deposit result in CPU register.
	XOR	CDST.SRC <, FD >	2	?	?	X	X	X	$[CDST] \leftarrow [SRC(FD)] \oplus [CDST]$ Exclusive-OR CPU register contents with any register's contents; deposit result in CPU register.
	CMPR	CDST.SRC <, FD >	2	X	X	X	X	X	$[SRC(FD)] - [CDST]$ Compare contents of CPU register with those of any register. Only the statuses are affected.
REGISTER OPERATE	ADDC	CREG	2	X	X	X	X	X	$[CREG] \leftarrow [CREG] + [C]$ Add Carry bit to contents of CPU register.
	NEG	CREG	2	X	X	X	X	X	$[CREG] \leftarrow 0 - [CREG]$ Negate contents of CPU register (twos complement).
	CMPL	CREG	2	?	?	X	X	X	$[CREG] \leftarrow [CREG]$ Complement contents of CPU register (ones complement).
	SHFTL	RRL, CREG <, C >	2	X	X	X	X	X	 <p>Rotate CPU register contents left one bit position, through Carry if specified.</p>
	SHFTL	LSL, CREG, F	2	X	X	X	X	X	 <p>Shift CPU register contents left one bit position, filling bit 0 according to F.</p>

Table 20-1. A Summary Of The MC2 Instruction Set (Continued)

TYPE	MNEMONIC	OPERAND(S)	BYTES	STATUSES				OPERATION PERFORMED
				O	C	N	Z	
REGISTER OPERATE (CONTINUED)	SHFTR	RRR,CREG,<C>	2	?	X	X	X	 <p>Rotate CPU register contents right one bit position, through Carry if specified.</p>
	SHFTR	LSR,CREG,F	2	?	X	X	X	 <p>Shift CPU register contents right one bit position, filling bit 15 according to F.</p>
I/O AND BIT MANIPULATION	SBIT	CDST,SRCK	2	?	?	X	X	$[SRC < K >] \leftarrow 1$ $[CDST] \leftarrow [SRC]$ Set the specified bit of the specified register to 1, then deposit result in a CPU register.
	RBIT	CDST,SRCK	2	?	?	X	X	$[SRC < K >] \leftarrow 0$ $[CDST] \leftarrow [SRC]$ Reset the specified bit of the specified register to 0, then deposit result in a CPU register.
	CBIT	CDST,SRCK	2	?	?	X	X	$[SRC < K >] \leftarrow [SRC < K >]$ $[CDST] \leftarrow [SRC]$ Complement the specified bit of the specified register, then deposit result in a CPU register.
	TBIT	CDST,SRCK	2	?	?	X	X	$[CDST] \leftarrow [SRC] \wedge 2^{**K}$ Set all bits of the specified register, except the specified bit, to 0; deposit result in a CPU register.
STACK	PUSH	CREG	2					$[[SP]] \leftarrow [CREG]$ $[SP] \leftarrow [SP] + 1$ Store CPU register's contents on top of stack.
	POP	CREG	2					$[CREG] \leftarrow [[SP] - 1]; [SP] \leftarrow [SP] - 1$ Load CPU register from top of stack.
	HALT		2					CPU enters idle state.