THE MC6809 INSTRUCTION SET

The following sections introduce the more commonly used MC6809 instructions and describe some typical uses for them.

1 The MC6809 Data Movement Instructions

This section includes instructions, which allow bit patterns to be moved without being modified in any way. It includes instructions for moving data within the processor as well as between memory and processor registers.

The basic instructions to move data between memory and the processor are **LD?** (load) and **ST?** (store), where **?** represents the processor register involved. Examples are **LDA, LDX, STD,** and **STS**. These instructions use any one of three of the MC6809 addressing modes: direct, extended, or indexed. The load instructions may also use the immediate mode. These instructions are used primarily to initialize registers and to store results into memory.

A useful variation of load/store is the clear instruction, which loads or stores a specific value of 0. This instruction clears every bit in a designated accumulator (**CLRA** or **CLRB**) or in a memory location (**CLR** followed by the appropriate operand). This instruction is used to initialize running sums to 0 or to establish starting values for other varying quantities.

The two instructions, exchange (**EXG**) and transfer (**TFR**), move data between a designated pair of registers. EXG exchanges the contents of the two registers; TFR transfers the content of the first register into the second register. Figure 3 compares the exchange and transfer operations.

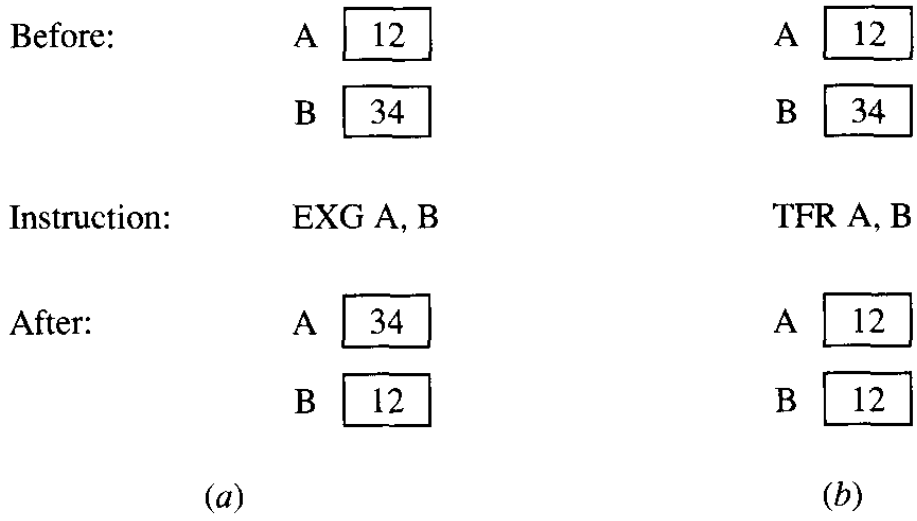

FIGURE 3 The MC6809 EXG and TFR instructions: (a) EXG, *(b)* TFR.

Load effective address (**LEA?**) is an instruction which calculates an address and then loads it into the double register (**S, U, X,** or **Y**) designated in the mnemonic. It must use the indexed addressing mode and one of several allowed options. When the effective address is calculated as called for in the option selected, it is not used to find an operand but is loaded into the double register. Figure 4 shows several examples of the LEA instruction.

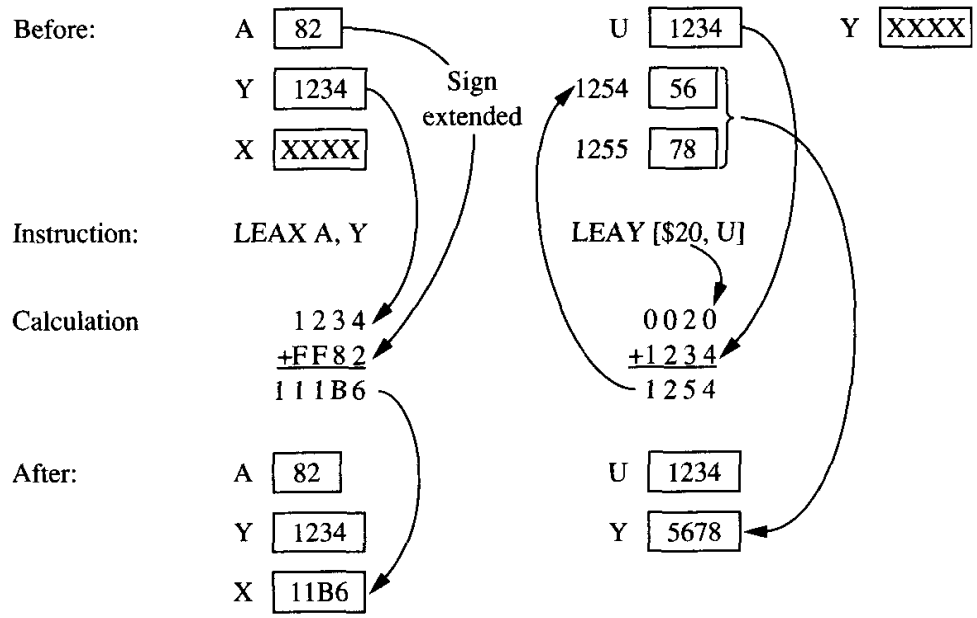

FIGURE 4 Examples of the MC6809 LEA instruction: (a) register offset, *(b)* indirect.

2 The MC6809 Arithmetic Instructions

These instructions perform arithmetic or arithmetic-related operations on the contents of registers or memory locations.

The basic arithmetic instructions add or subtract the content of a memory location to or from the content of a specified accumulator. The mnemonics are **ADD**? and **SUB**?, where the designation may be **A, B**, or **D**. These instructions use any one of four of the MC6809 addressing modes: immediate, direct, extended, or indexed.

Other variants add 1 to (increment, **INC**) or subtract 1 from (decrement, **DEC**) the value in the designated accumulator or the value in a memory location identified in one of the addressing modes.

Examples are **INCA, DEC 1234H,** and **INC ,X**.

These instructions are used primarily to increment counters used in a program. Although there is no counterpart instruction to increment any of the double registers, the **LEA** instruction may be used to do so. For example, **LEAX -1,X** will decrement register **X**.

Several instructions include the carry bit in the arithmetic in order to make it possible to extend the precision of the machine beyond eight bits. The add and subtract with carry instructions (**ADCA, ADCB, SBCA**, and **SBCB**) add or subtract the content of the specified memory location to or from the content of the designated accumulator. They then add to or subtract from that result the content of the carry flag in the condition codes register. This instruction supports multi-precision operations by providing a means to couple to a higher byte-result the carry from a lower byte-result of a multi-precision operation. Figure 5 shows an example of double-precision addition using the ADC instruction.

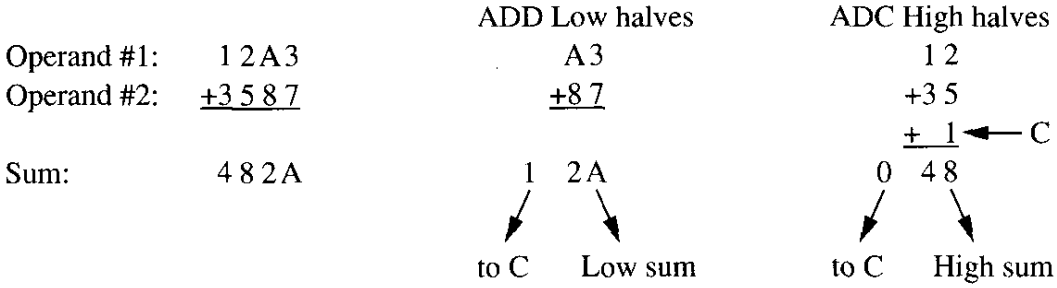

FIGURE 5 Double-precision addition in the MC6809.

Figure 6 shows an example of double-precision subtraction.

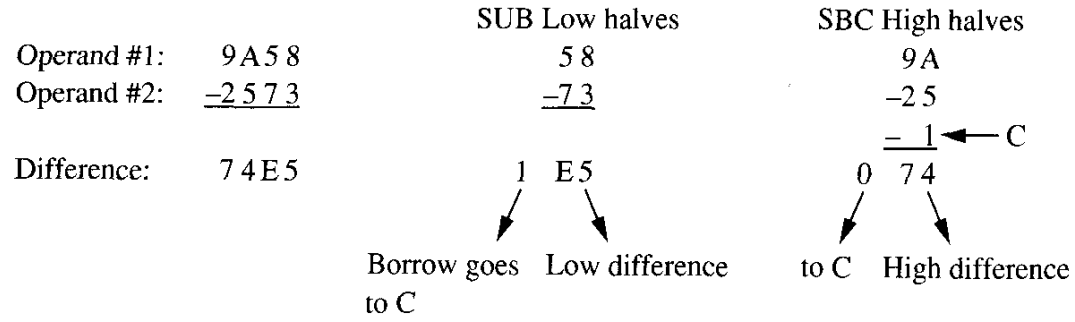

FIGURE 6 Double-precision subtraction in the MC6809.

The compare instruction (**CMP**?) subtracts a value in the memory location identified in one of the addressing modes from the designated register (**A, B, D, S, U, X,** or **Y**), shown here as **?**, but then discards the result after using it to update the condition code bits. This instruction is used to make numerical comparisons of the register content with the value in memory without changing the value in the register.

A similar instruction is the test instruction (**TSTA, TSTB,** or **TST** followed by an addressed operand), which subtracts 0 from the value in a designated location or accumulator. It is used to update the condition code register bits to reflect the actual target value, whether it is positive or negative, zero or nonzero.

The negate instruction (**NEGA, NEGB,** or **NEG** operand) changes the sign (2's complement) of the value in the designated accumulator or memory location.

The sign-extend instruction, **SEX**, sign-extends the value of accumulator B into accumulator A as shown in Figure 7. It converts an 8-bit signed number in B into a 16-bit signed number in D (A concatenated with B) by copying the sign bit from B into every bit position in A.

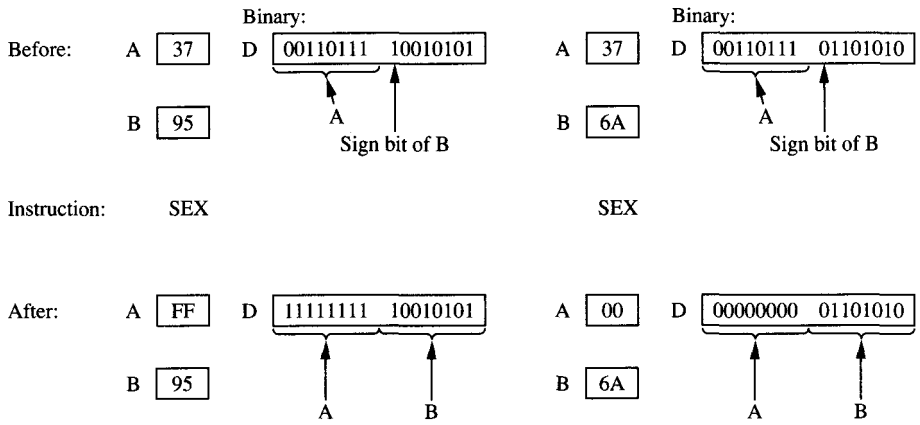

FIGURE 7 The MC6809 sign-extend instruction.

The MC6809 includes a single unsigned multiply instruction (**MUL**) which multiplies the content of accumulator A by the content of accumulator B and loads the 16-bit product into the double register D (A concatenated with B). It uses the inherent mode and has no other options as to operands or destination.

The instructions "arithmetic shift left" (**ASL**) and "arithmetic shift right" (**ASR**) can be used to perform simple multiply and divide operations. **ASL** shifts left by one bit the pattern while bringing a 0 in from the right. The leftmost bit shifts into the carry as shown in Figure 8. This operation results in doubling the value.

Arithmetic shift right (**ASR**) shifts the pattern right by one bit while bringing a copy of bit 7 (the sign bit) into the leftmost position.

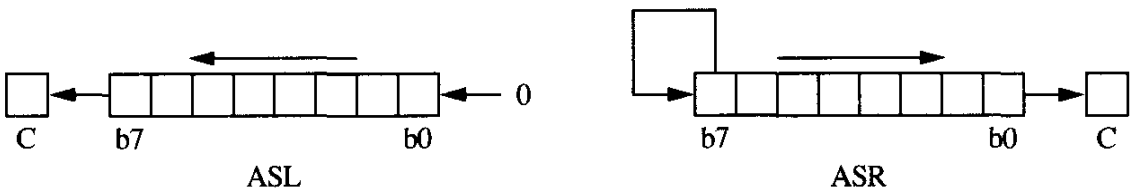

FIGURE 8 The MC6809 arithmetic shifts.

3 The MC6809 Logic Instructions

The instructions listed in this section perform logic operations or other bit-level manipulations on the contents of registers or memory locations. Any of three logic operations, **AND, OR,** and **EOR** (exclusive-OR), may be performed between the individual bits of the specified accumulator and those of the content of an identified memory location. The corresponding mnemonics are **ANDA, ANDB, ORA, ORB, EORA,** and **EORB**.

The **AND** operation is commonly used to ***clear*** certain bits in the accumulator or the CC register (those bits in positions where the other operand contains a 0). The **OR** operation is used to ***set*** certain bits, those in positions where the other operand contains a 1. The **EOR** operation ***complements*** selected bits in an accumulator, namely those in bit positions where the other operand contains a 1. Figure 9 illustrates these basic logic operations.

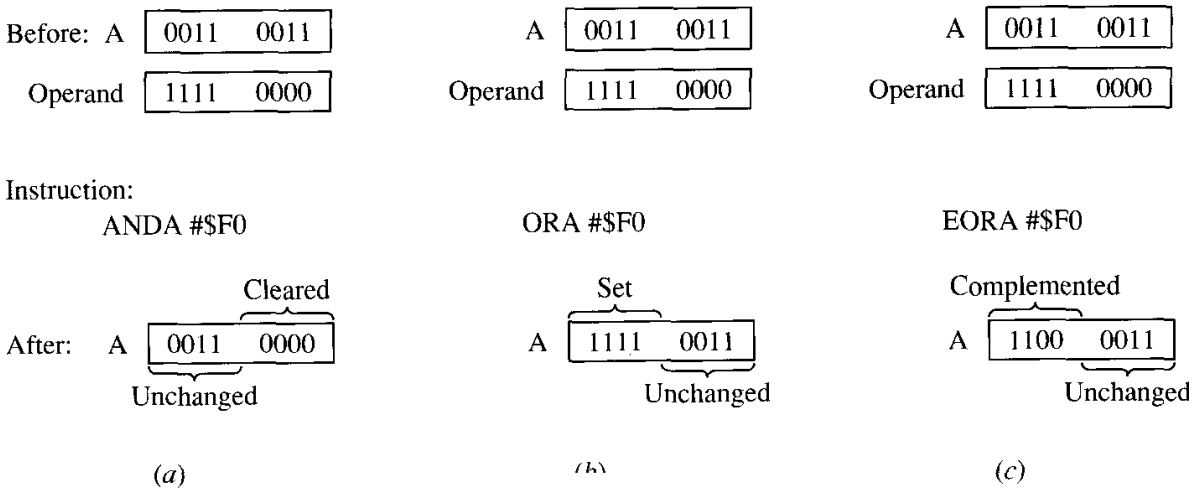

FIGURE 9 The basic MC6809 logic operations: *(a)* AND, *(b}* OR, (c) *(EOR).*

The logic shift left instruction (**LSL**) is identical to the arithmetic shift left (**ASL**). The logic shift/rotate instructions operate as shown in Figure 10

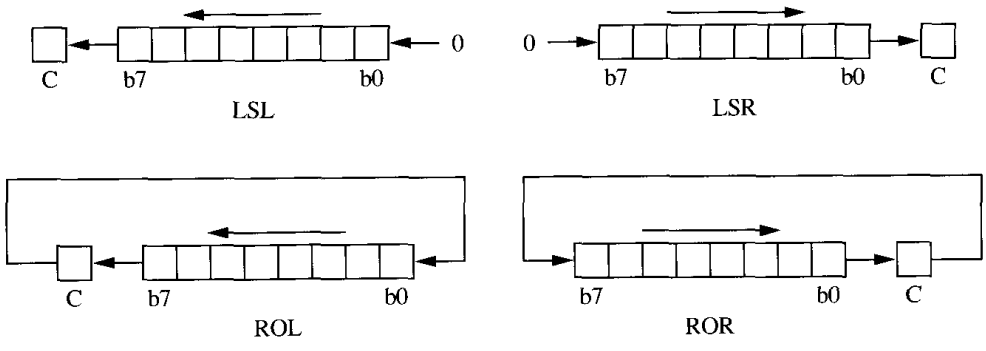

FIGURE 10 The MC6809 logic shift/rotate instructions.

4 The MC6809 Test/Branch Instructions

This section describes those instructions which may cause the sequence of execution to branch to a different region of the program. The majority of these instructions in the MC6809 use the relative addressing mode: the address to which the program may branch is included as an offset from the next sequential instruction after the branch.

The "branch" instructions use the relative addressing mode. The offset specified in the machine code may be a single-byte or a double-byte. The latter is indicated in the source program by the initial letter of L (long) in the mnemonic. In assembly language it is most convenient to write the destination address of a branch as a label.

A ***conditional branch*** is taken if the condition is satisfied. The decision is based upon the value of certain bits in the condition code register at the time the branch instruction is executed. Some of the numerical conditions (greater than, less than or equal, and so forth) refer to the result of a preceding subtraction (or compare) instruction in which the content of a memory location has been subtracted from the content of an accumulator. The result of such an operation must be interpreted ***differently*** depending on whether the patterns are assumed to represent ***signed or unsigned numbers***.

The MC6809 branch instructions, each with a long and a short form, are listed in Table. Each word description should be preceded by the words "branch if," and the branch will be taken if the result of the test listed in the table is true. The symbols *r* and *m* refer to a preceding subtraction or compare instruction in which the content of a memory location *(m)* is subtracted from the content of an accumulator (r). The results of the number tests involving *r* and *m* will be properly determined even in cases where the result itself may overflow the machine, that is, the tests involve the appropriate carry or overflow bits.

In addition to the "branch" instructions, the MC6809 includes an unconditional branch instruction called the jump (**JMP**) instruction. This instruction causes a branch in the execution of the program to the location whose address is identified in one of three addressing modes: direct, extended, or indexed. This branch instruction is used whenever the programmer wishes to use one of these addressing modes to identify the target address for an unconditional branch.

The branch never instruction (**BRN**) may be used to reserve a place for a branch to be added to a program at some future time or to temporarily replace a branch in order to facilitate the debugging of a program. The branch always (**BRA**) and the branch to subroutine (**BSR**) are unconditional branches.

The MC6809 branch instructions

Instruction	Word description Branch if:	Test
(L)BCC	carry bit is clear	C = 0
(L)BCS	carry bit is set	C = 1
(L)BEQ	equal (to 0)	r = m(Z =1)
(L)BGE	greater than or equal	r > m (signed)
(L)BGT	greater than	r > m (signed)
(L)BHI	higher than	r > m (unsigned)
(L)BHS	higher than or the same	r > m (unsigned)
(L)BLE	less than or equal	r < m (signed)
(L)BLO	lower than	r < m (unsigned)
(L)BLS	lower than or the same	r < m (unsigned)
(L)BLT	less than	r < m (signed)
(L)BMI	minus	N = 1
(L)BNE	not equal (to 0)	r ≠ m (Z = 0)
(L)BPL	plus	N = 0
(L)BRA	always	always branch
(L)BRN	never	never branch
(L)BSR	always (branch to subroutine)	always
(L)BVC	no overflow	V = 0
(L)BVS	overflow	V - 1

5 The MC6809 No-operation Instruction

An instruction available with most microprocessors is one which does nothing. In the MC6809 it is the no-operation instruction (NOP). When it is executed it causes no activity other than the normal PC increment to take place.