Nomenclature

The following nomenclature is used in the subsequent definitions.

1. Operators:
   - ( ) = Contents of register shown inside parentheses
   - ⇐ = Is transferred to
   - ⇑ = Is pulled from stack
   - ⇓ = Is pushed onto stack
   - • = Boolean AND
   - + = Arithmetic addition symbol except where used as inclusive-OR symbol in Boolean formula
   - ⊕ = Exclusive OR
   - × = Multiply
   - : = Concatenation
   - – = Arithmetic subtraction symbol or negation symbol (twos complement)
2. Registers in the CPU:
   - ACCA = Accumulator A
   - ACCB = Accumulator B
   - ACCX = Accumulator ACCA or ACCB
   - ACCD = Double accumulator — Accumulator A concatenated with accumulator B where A is the most significant byte
   - CCR = Condition code register
   - IX = Index register X, 16 bits
   - IXH = Index register X, high order 8 bits
   - IXL = Index register X, low order 8 bits
   - PC = Program counter, 16 bits
   - PCH = Program counter, high order (most significant) 8 bits
   - PC = Program counter, low order (least significant) 8 bits
   - SP = Stack pointer, 16 bits
   - SPH = Stack pointer, high order 8 bits
   - SPL = Stack pointer, low order 8 bits

3. Memory and addressing:
   - M = A memory location (one byte)
   - M+1 = The byte of memory at $0001 plus the address of the memory location indicated by “M”
   - Rel = Relative offset (that is, the twos complement number stored in the last byte of machine code corresponding to a branch instruction)
   - (opr) = Operand
   - (msk) = Mask used in bit manipulation instructions
   - (rel) = Relative offset used in branch instructions

4. Bits [7:0] of the condition code register:
   - S = Stop disable, bit 7
   - X = X interrupt mask, bit 6
   - H = Half carry, bit 5
   - I = I interrupt mask, bit 4
   - N = Negative indicator, bit 3
   - Z = Zero indicator, bit 2
   - V = Two’s complement overflow indicator, bit 1
   - C = Carry/borrow, bit 0

5. Status of individual bit before execution of an instruction:
   - An = Bit n of ACCA (n = 7, 6, 5... 0)
   - Bn = Bit n of ACCB (n = 7, 6, 5... 0)
   - Dn = Bit n of ACCD (n = 15, 14, 13... 0) where bits [15:8] refer to ACCA and bits [7:0] refer to ACCB
   - IXn = Bit n of IX (n = 15, 14, 13... 0)
   - IXHn = Bit n of IXH (n = 7, 6, 5... 0)
   - IXLn = Bit n of IXL (n = 7, 6, 5... 0)
   - IYn = Bit n of IY (n = 15, 14, 13... 0)
   - IYHn = Bit n of IYH (n = 7, 6, 5... 0)
   - IYLn = Bit n of IYL (n = 7, 6, 5... 0)
   - Mn = Bit n of M (n = 7, 6, 5... 0)
   - SPHn = Bit n of SPH (n = 7, 6, 5... 0)
   - SPLn = Bit n of SPL (n = 7, 6, 5... 0)
   - Xn = Bit n of ACCX (n = 7, 6, 5... 0)

6. Status of individual bits of result of execution of an instruction:

   For 8-bit results:

   Rn = Bit n of the result (n = 7, 6, 5... 0). This applies to instructions which provide a result contained in a single byte of memory or in an 8-bit register.

   For 16-bit results:

   RHn = Bit n of the most significant byte of the result (n = 7, 6, 5... 0)

   RLn = Bit n of the least significant byte of the result (n = 7, 6, 5... 0). This applies to instructions which provide a result contained in two consecutive bytes of memory or in a 16-bit register.

   Rn = Bit n of the result (n = 15, 14, 13... 0)

7. Notation used in CCR activity summary figures:

   — = Bit not affected

   0 = Bit forced to 0

   1 = Bit forced to 1

   ⇳ = Bit set or cleared according to results of operation

   ⇓ = Bit may change from 1 to 0, remain 0, or remain 1 as a result of this operation, but cannot change from 0 to 1.

8. Notation used in cycle-by-cycle execution tables:

   — = Irrelevant data

   ii = One byte of immediate data

   jj = High-order byte of 16-bit immediate data

   kk = Low-order byte of 16-bit immediate data

   hh = High-order byte of 16-bit extended address

   ll = Low-order byte of 16-bit extended address

   dd = Low-order eight bits of direct address $0000–$00FF (high byte assumed to be $00)

   mm = 8-bit mask (set bits correspond to operand bits which will be affected)

   ff = 8-bit forward offset $00 (0) to $FF (255) (is added to index)

rr = Signed relative offset $80 (–128) to $7F (+127) (offset relative to address following machine code offset byte)
OP = Address of opcode byte
OP+n = Address of n^{th} location after opcode byte
SP = Address pointed to by stack pointer value (at the start of an instruction)
SP+n = Address of n^{th} higher address past that pointed to by stack pointer
SP–n = Address of n^{th} lower address before that pointed to by stack pointer
Sub = Address of called subroutine
Nxt op = Opcode of next instruction
Rtn hi = High-order byte of return address
Rtn lo = Low-order byte of return address
Svc hi = High-order byte of address for service routine
Svc lo = Low-order byte of address for service routine
Vec hi = High-order byte of interrupt vector
Vec lo = Low-order byte of interrupt vector

M6803 Instruction Set

The instructions are arranged in alphabetical order with the instruction mnemonic set in larger type for easy reference.

ABA — Add Accumulator B to Accumulator A — ABA

Operation: ACCA ⇐ (ACCA) + (ACCB)

Description: Adds the contents of accumulator B to the contents of accumulator A and places the result in accumulator A. Accumulator B is not changed. This instruction affects the H condition code bit so it is suitable for use in BCD arithmetic operations (see **DAA** instruction for additional information).

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	⇕	—	⇕	⇕	⇕	⇕

H $A3 \bullet B3 + B3 \bullet \overline{R3} + \overline{R3} \bullet A3$
Set if there was a carry from bit 3; cleared otherwise.

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $A7 \bullet B7 \bullet \overline{R7} + \overline{A7} \bullet \overline{B7} \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $A7 \bullet B7 + B7 \bullet \overline{R7} + \overline{R7} \bullet A7$
Set if there was a carry from the MSB of the result; cleared otherwise.

Source Form: ABA

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ABA (INH)		
	Addr	Data	R/W
1	OP	1B	1
2	OP + 1	—	1

ABX — Add Accumulator B to Index Register X — ABX

Operation: IX ⇐ (IX) + (ACCB)

Description: Adds the 8-bit unsigned contents of accumulator B to the contents of index register X (IX) considering the possible carry out of the low-order byte of the index register X; places the result in index register X (IX). Accumulator B is not changed. There is no equivalent instruction to add accumulator A to an index register.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: ABX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ABX (INH)		
	Addr	Data	R/W
1	OP	3A	1
2	OP + 1	—	1
3	FFFF	—	1

ADC — Add with Carry — ADC

Operation: ACCX ⇐ (ACCX) + (M) + (C)

Description: Adds the contents of the C bit to the sum of the contents of ACCX and M and places the result in ACCX. This instruction affects the H condition code bit so it is suitable for use in BCD arithmetic operations (see **DAA** instruction for additional information).

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	⇕	—	⇕	⇕	⇕	⇕

H $X3 \bullet M3 + M3 \bullet \overline{R3} + \overline{R3} \bullet X3$
Set if there was a carry from bit 3; cleared otherwise.

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $X7 \bullet M7 \bullet \overline{R7} + \overline{X7} \bullet \overline{M7} \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $X7 \bullet M7 + M7 \bullet \overline{R7} + \overline{R7} \bullet X7$
Set if there was a carry from the MSB of the result; cleared otherwise.

Source Forms: ADCA (opr); ADCB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ADCA (IMM)			ADCA (DIR)			ADCA (EXT)			ADCA (IND,X)			ADCA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	89	1	OP	99	1	OP	B9	1	OP	A9	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A9	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X+ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	ADCB (IMM)			ADCB (DIR)			ADCB (EXT)			ADCB (IND,X)			ADCB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C9	1	OP	D9	1	OP	F9	1	OP	E9	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E9	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

ADD — Add without Carry — ADD

Operation: ACCX ⇐ (ACCX) + (M)

Description: Adds the contents of M to the contents of ACCX and places the result in ACCX. This instruction affects the H condition code bit so it is suitable for use in BCD arithmetic operations (see **DAA** instruction for additional information).

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	⇕	—	⇕	⇕	⇕	⇕

H $X3 \bullet M3 + M3 \bullet \overline{R3} + \overline{R3} \bullet X3$
Set if there was a carry from bit 3; cleared otherwise.

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $X7 \bullet M7 \bullet \overline{R7} + \overline{X7} \bullet \overline{M7} \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $X7 \bullet M7 + M7 \bullet \overline{R7} + \overline{R7} \bullet X7$
Set if there was a carry from the MSB of the result; cleared otherwise.

Source Forms: ADDA (opr); ADDB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ADCA (IMM)			ADCA (DIR)			ADCA (EXT)			ADCA (IND,X)			ADCA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	8B	1	OP	9B	1	OP	BB	1	OP	AB	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AB	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	ADDB (IMM)			ADDB (DIR)			ADDB (EXT)			ADDB (IND,X)			ADDB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	CB	1	OP	DB	1	OP	FB	1	OP	EB	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EB	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

ADDD — Add Double Accumulator — ADDD

Operation: ACCD ⇐ (ACCD) + (M : M + 1)

Description: Adds the contents of M concatenated with M + 1 to the contents of ACCD and places the result in ACCD. Accumulator A corresponds to the high-order half of the 16-bit double accumulator D.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

V $D15 \bullet M15 \bullet \overline{R15} + \overline{D15} \bullet \overline{M15} \bullet R15$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $D15 \bullet M15 + M15 \bullet \overline{R15} + \overline{R15} \bullet D15$
Set if there was a carry from the MSB of the result; cleared otherwise.

Source Form: ADDD (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ADDD (IMM)			ADDD (DIR)			ADDD (EXT)			ADDD (IND,X)			ADDD (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C3	1	OP	D3	1	OP	F3	1	OP	E3	1	OP	18	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E3	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	FFFF	—	1	00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5				FFFF	—	1	hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6							FFFF	—	1	FFFF	—	1	Y + ff + 1	(Y + ff + 1)	1
7													FFFF	—	1

AND — Logical AND — AND

Operation: ACCX ⇐ (ACCX) • (M)

Description: Performs the logical AND between the contents of ACCX and the contents of M and places the result in ACCX. (Each bit of ACCX after the operation will be the logical AND of the corresponding bits of M and of ACCX before the operation.)

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇳	⇳	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

Source Forms: ANDA (opr); ANDB (opr)

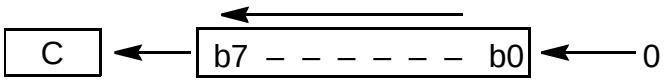
Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ADCA (IMM)			ADCA (DIR)			ADCA (EXT)			ADCA (IND,X)			ADCA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	84	1	OP	94	1	OP	B4	1	OP	A4	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A4	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	ANDB (IMM)			ANDB (DIR)			ANDB (EXT)			ANDB (IND,X)			ANDB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C4	1	OP	D4	1	OP	F4	1	OP	E4	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E4	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

ASL — Arithmetic Shift Left (Same as LSL) — ASL

Operation:



Description: Shifts all bits of the ACCX or M one place to the left. Bit 0 is loaded with a 0. The C bit in the CCR is loaded from the most significant bit of ACCX or M.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the shift).

C M7
Set if, before the shift, the MSB of ACCX or M was set; cleared otherwise.

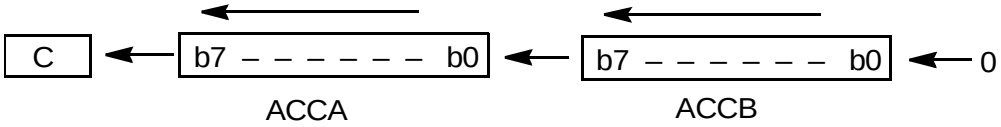
Source Forms: ASLA; ASLB; ASL (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ASLA (IMM)			ASLB (DIR)			ASL (EXT)			ASL (IND,X)			ASL (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	48	1	OP	58	1	OP	78	1	OP	68	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	68	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

ASLD — Arithmetic Shift Left Double Accumulator (Same as LSLD) — ASLD

Operation:



Description: Shifts all bits of ACCD one place to the left. Bit 0 is loaded with a 0. The C bit in the CCR is loaded from the most significant bit of ACCD.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the shift).

C D15
Set if, before the shift, the MSB of ACCD was set; cleared otherwise.

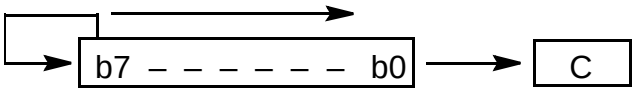
Source Form: ASLD

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ASLD (INH)		
	Addr	Data	R/W
1	OP	05	1
2	OP + 1	—	1
3	FFFF	—	1

ASR — Arithmetic Shift Right — ASR

Operation:



Description: Shifts all bits of the ACCX or M one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C bit of the CCR. This operation effectively divides a twos complement value by two without changing its sign. The carry bit can be used to round the result.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the shift).

C M0
Set if, before the shift, the LSB of ACCX or M was set; cleared otherwise.

Source Forms: ASRA; ASRB; ASR (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ASRA (IMM)			ASRB (DIR)			ASR (EXT)			ASR (IND,X)			ASR (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	47	1	OP	57	1	OP	77	1	OP	67	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	67	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

BCC

Branch if Carry Clear (Same as BHS)

BCC

Operation: PC ⇐ (PC) + $0002 + Rel if (C) = 0

Description: Tests the state of the C bit in the CCR and causes a branch if C is clear. See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BCC (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BCC (REL)		
	Addr	Data	R/W
1	OP	24	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BCS

Branch if Carry Set (Same as BLO)

BCS

Operation: PC ⇐ (PC) + $0002 + Rel if (C) = 1

Description: Tests the state of the C bit in the CCR and causes a branch if C is set. See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BCS (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BCS (REL)		
	Addr	Data	R/W
1	OP	25	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BEQ — Branch if Equal — BEQ

Operation: PC ⇐ (PC) + $0002 + Rel if (Z) = 1

Description: Tests the state of the Z bit in the CCR and causes a branch if Z is set. See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BEQ (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BEQ (REL)		
	Addr	Data	R/W
1	OP	27	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BGE — Branch if Greater than or Equal to Zero — BGE

Operation: PC ⇐ (PC) + $0002 + Rel if (N) ⊕ (V) = 0
i.e., if (ACCX) ≥ (M) (twos-complement signed numbers)

Description: If the BGE instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the twos complement number represented by ACCX was greater than or equal to the twos complement number represented by M.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BGE (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BGE (REL)		
	Addr	Data	R/W
1	OP	2C	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BGT — Branch if Greater than Zero — BGT

Operation: PC ⇐ (PC) + $0002 + Rel if (Z) + [(N) ⊕ (V)] = 0
i.e., if (ACCX) > (M) (twos-complement signed numbers)

Description: If the BGT instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the twos complement number represented by ACCX was greater than the twos complement number represented by M.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BGT (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BGT (REL)		
	Addr	Data	R/W
1	OP	2E	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BHI — Branch if Higher — BHI

Operation: PC ⇐ (PC) + $0002 + Rel if (C) + (Z) = 0
i.e., if (ACCX) > (M) (unsigned binary numbers)

Description: If the BHI instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the unsigned binary number represented by ACCX was greater than unsigned binary number represented by M. Generally not useful after INC/DEC, LD/ST, TST/CLR/COM because these instructions do not affect the C bit in the CCR.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BHI (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BHI (REL)		
	Addr	Data	R/W
1	OP	22	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BHS Branch if Higher or Same (Same as BCC) BHS

Operation: PC ⇐ (PC) + $0002 + Rel if (C) = 0
i.e., if (ACCX) ≥ (M) (unsigned binary numbers)

Description: If the BHS instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the unsigned binary number represented by ACCX was greater than or equal to the unsigned binary number represented by M. Generally not useful after INC/DEC, LD/ST, TST/CLR/COM because these instructions do not affect the C bit in the CCR.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BHS (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BHS (REL)		
	Addr	Data	R/W
1	OP	24	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BIT — Bit Test — BIT

Operation: (ACCX) • (M)

Description: Performs the logical AND between the contents of ACCX and the contents of M and modifies the condition codes accordingly. Neither the contents of ACCX nor M operands are affected. (Each bit of the result of the AND would be the logical AND of the corresponding bits of ACCX and M.)

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

Source Forms: BITA (opr); BITB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BITA (IMM)			BITA (DIR)			BITA (EXT)			BITA (IND,X)			BITA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	85	1	OP	95	1	OP	B5	1	OP	A5	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AS	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	BITB (IMM)			BITB (DIR)			BITB (EXT)			BITB (IND,X)			BITB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C5	1	OP	D5	1	OP	F5	1	OP	E5	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	ES	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

BLE — Branch if Less than or Equal to Zero — BLE

Operation: PC ⇐ (PC) + $0002 + Rel if (Z) + [(N) ⊕ (V)] = 1
i.e., if (ACCX) ≤ (M) (twos complement signed numbers)

Description: If the BLE instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the twos complement signed number represented by ACCX was less than or equal to the twos complement signed number represented by M.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BLE (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BLE (REL)		
	Addr	Data	R/W
1	OP	2F	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BLO — Branch if Lower (Same as BCS) — BLO

Operation: PC ⇐ (PC) + $0002 + Rel if (C) = 1
i.e., if (ACCX) < (M) (unsigned binary numbers)

Description: If the BLO instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the unsigned binary number represented by ACCX was less than the unsigned binary number represented by M. Generally not useful after INC/DEC, LD/ST, TST/CLR/COM because these instructions do not affect the C bit in the CCR.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BLO (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BLO (REL)		
	Addr	Data	R/W
1	OP	25	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BLS — Branch if Lower or Same — BLS

Operation:

PC ⇐ (PC) + $0002 + Rel if (C) + (Z) = 1
i.e., if (ACCX) ≤ (M) (unsigned binary numbers)

Description:

If the BLS instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the unsigned binary number represented by ACCX was less than or equal to the unsigned binary number represented by M. Generally not useful after INC/DEC, LD/ST, TST/CLR/COM because these instructions do not affect the C bit in the CCR.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form:

BLS (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BLS (REL)		
	Addr	Data	R/W
1	OP	23	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BLT — Branch if Less than Zero — BLT

Operation: PC ⇐ (PC) + $0002 + Rel if (N) ⊕ (V)= 1
i.e., if (ACCX) < (M) (twos complement signed numbers)

Description: If the BLT instruction is executed immediately after execution of any of the instructions, CBA, CMP(A, B, or D), CP(X or Y), SBA, SUB(A, B, or D), the branch will occur if and only if the twos-complement number represented by ACCX was less than the twos-complement number represented by M.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BLT (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BLT (REL)		
	Addr	Data	R/W
1	OP	2D	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BMI — Branch if Minus — BMI

Operation: PC ⇐ (PC) + $0002 + Rel if (N) = 1

Description: Tests the state of the N bit in the CCR and causes a branch if N is set. See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BMI (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BMI (REL)		
	Addr	Data	R/W
1	OP	2B	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BNE — Branch if Not Equal to Zero — BNE

Operation: PC ⇐ (PC) + $0002 + Rel if (Z) = 0

Description: Tests the state of the Z bit in the CCR and causes a branch if Z is clear. See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BNE (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BNE (REL)		
	Addr	Data	R/W
1	OP	26	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BPL — Branch if Plus — BPL

Operation: PC ⇐ (PC) + $0002 + Rel if (N) = 0

Description: Tests the state of the N bit in the CCR and causes a branch if N is clear. See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BPL (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BPL (REL)		
	Addr	Data	R/W
1	OP	2A	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BRA — Branch Always — BRA

Operation: PC ⇐ (PC) + $0002 + Rel

Description: Unconditional branch to the address given by the foregoing formula, in which Rel is the relative offset stored as a twos-complement number in the second byte of machine code corresponding to the branch instruction.

The source program specifies the destination of any branch instruction by its absolute address, either as a numerical value or as a symbol or expression, that can be numerically evaluated by the assembler. The assembler obtains the relative address, Rel, from the absolute address and the current value of the location counter.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BRA (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BRA (REL)		
	Addr	Data	R/W
1	OP	20	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BRN — Branch Never — BRN

Operation: PC ⇐ (PC) + $0002

Description: Never branches. In effect, this instruction can be considered as a 2-byte NOP (no operation) requiring three cycles for execution. Its inclusion in the instruction set is to provide a complement for the BRA instruction. This instruction is useful during program debug to negate the effect of another branch instruction without disturbing the offset byte. Having a complement for BRA is also useful in compiler implementations.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BRN (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BRN (REL)		
	Addr	Data	R/W
1	OP	21	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BSR — Branch to Subroutine — BSR

Operation:

PC ⇐ (PC) + $0002	Advance PC to return address
⇓ (PCL)	Push low-order return onto stack
SP ⇐ (SP) – $0001	
⇓ (PCH)	Push high-order return onto stack
SP ⇐ (SP) – $0001	
PC ⇐ (PC) + Rel	Load start address of requested subroutine

Description: The program counter is incremented by two (this will be the return address). The least significant byte of the contents of the program counter (low-order return address) is pushed onto the stack. The stack pointer is then decremented by one. The most significant byte of the contents of the program counter (high-order return address) is pushed onto the stack. The stack pointer is then decremented by one. A branch then occurs to the location specified by the branch offset.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BSR (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BSR (REL)		
	Addr	Data	R/W
1	OP	8D	1
2	OP + 1	rr	1
3	FFFF	—	1
4	Sub	Nxt op	1
5	SP	Rtn lo	0
6	SP–1	Rtn hi	0

BVC — Branch if Overflow Clear — BVC

Operation: PC ⇐ (PC) + $0002 + Rel if (V) = 0

Description: Tests the state of the V bit in the CCR and causes a branch if V is clear. Used after an operation on twos-complement binary values, this instruction will cause a branch if there was NO overflow. That is, branch if the twos-complement result was valid.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BVC (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BVC (REL)		
	Addr	Data	R/W
1	OP	28	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

BVS — Branch if Overflow Set — BVS

Operation: PC ⇐ (PC) + $0002 + Rel if (V) = 1

Description: Tests the state of the V bit in the CCR and causes a branch if V is set. Used after an operation on twos-complement binary values, this instruction will cause a branch if there was an overflow. That is, branch if the twos-complement result was invalid.

See **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: BVS (rel)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	BVS (REL)		
	Addr	Data	R/W
1	OP	29	1
2	OP + 1	rr	1
3	FFFF	—	1

The following table is a summary of all branch instructions.

Test	Boolean	Mnemonic	Opcode	Complementary		Branch	Comment
r > m	Z + (N ⊕ V) = 0	BGT	2E	r ≤ m	BLE	2F	Signed
r ≥ m	N ⊕ V = 0	BGE	2C	r < m	BLT	2D	Signed
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Signed
r ≤ m	Z + (N ⊕ V) = 1	BLE	2F	r > m	BGT	2E	Signed
r < m	N ⊕ V = 1	BLT	2D	r ≥ m	BGE	2C	Signed
r > m	C + Z = 0	BHI	22	r ≤ m	BLS	23	Unsigned
r ≥ m	C = 0	BHS/BCC	24	r < m	BLO/BCS	25	Unsigned
r = m	Z = 1	BEQ	27	r ≠ m	BNE	26	Unsigned
r ≤ m	C + Z = 1	BLS	23	r > m	BHI	22	Unsigned
r < m	C = 1	BLO/BCS	25	r ≥ m	BHS/BCC	24	Unsigned
Carry	C = 1	BCS	25	No Carry	BCC	24	Simple
Negative	N = 1	BMI	2B	Plus	BPL	2A	Simple
Overflow	V = 1	BVS	29	No Overflow	BVC	28	Simple
r = 0	Z = 1	BEQ	27	r ≠ 0	BNE	26	Simple
Always	—	BRA	20	Never	BRN	21	Unconditional

CBA — Compare Accumulators — CBA

Operation: (ACCA) – (ACCB)

Description: Compares the contents of ACCA to the contents of ACCB and sets the condition codes, which may be used for arithmetic and logical conditional branches. Both operands are unaffected.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $A7 \bullet \overline{B7} \bullet \overline{R7} + \overline{A7} \bullet B7 \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{A7} \bullet B7 + B7 \bullet R7 + R7 \bullet \overline{A7}$
Set if there was a borrow from the MSB of the result; cleared otherwise.

Source Form: CBA

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CBA (INH)		
	Addr	Data	R/W
1	OP	11	1
2	OP + 1	—	1

CLC — Clear Carry — CLC

Operation: C bit ⇐ 0

Description: Clears the C bit in the CCR.

CLC may be used to set up the C bit prior to a shift or rotate instruction involving the C bit.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	0

C 0
Cleared

Source Form: CLC

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CLC (INH)		
	Addr	Data	R/W
1	OP	0C	1
2	OP + 1	—	1

CLI — Clear Interrupt Mask — CLI

Operation: I bit ⇐ 0

Description: Clears the interrupt mask bit in the CCR. When the I bit is clear, interrupts are enabled. There is one E-clock cycle delay in the clearing mechanism for the I bit so that, if interrupts were previously disabled, the next instruction after a CLI will always be executed, even if there was an interrupt pending prior to execution of the CLI instruction.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	0	—	—	—	—

I 0
Cleared

Source Form: CLI

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CLI (INH)		
	Addr	Data	R/W
1	OP	0E	1
2	OP + 1	—	1

CLR — Clear — CLR

Operation: ACCX ⇐ 0 **or:** M ⇐ 0

Description: The contents of ACCX or M are replaced with 0s.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	0	1	0	0

N 0
Cleared

Z 1
Set

V 0
Cleared

C 0
Cleared

Source Forms: CLRA; CLRB; CLR (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CLRA (IMM)			CLRB (DIR)			CLR (EXT)			CLR (IND,X)			CLR (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	4F	1	OP	5F	1	OP	7F	1	OP	6F	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	6F	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	00	0	X + ff	00	0	FFFF	—	1
7													Y + ff	00	0

CLV — Clear Twos Complement Overflow Bit — CLV

Operation: V bit ⇐ 0

Description: Clears the twos complement overflow bit in the CCR

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	0	—

V 0
Cleared

Source Form: CLV

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CLV (INH)		
	Addr	Data	R/W
1	OP	0A	1
2	OP + 1	—	1

CMP — Compare — CMP

Operation: (ACCX) – (M)

Description: Compares the contents of ACCX to the contents of M and sets the condition codes, which may be used for arithmetic and logical conditional branching. Both operands are unaffected.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $X7 \bullet \overline{M7} \bullet \overline{R7} + \overline{X7} \bullet M7 \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{X7} \bullet M7 + M7 \bullet R7 + R7 \bullet \overline{X7}$
Set if there was a borrow from the MSB of the result; cleared otherwise.

Source Forms: CMPA (opr); CMPB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CMPA (IMM)			CMPA (DIR)			CMPA (EXT)			CMPA (IND,X)			CMPA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	81	1	OP	91	1	OP	B1	1	OP	A1	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A1	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	CMPB (IMM)			CMPB (DIR)			CMPB (EXT)			CMPB (IND,X)			CMPB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C1	1	OP	D1	1	OP	F1	1	OP	E1	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E1	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

COM — Complement — COM

Operation: ACCX ⇐ ($\overline{\text{ACCX}}$) = $FF – (ACCX) **or:** M ⇐ ($\overline{\text{M}}$) = $FF – (M)

Description: Replaces the contents of ACCX or M with its one's complement. (Each bit of the contents of ACCX or M is replaced with the complement of that bit.) To complement a value without affecting the C bit, EXclusive-OR the value with $FF.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	1

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

C 1
Set (For compatibility with M6800)

Source Forms: COMA; COMB; COM (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	COMA (INH)			COMB (INH)			COM (EXT)			COM (IND,X)			COM (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	43	1	OP	53	1	OP	73	1	OP	63	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	63	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

CPX — Compare Index Register X — CPX

Operation: (IX) – (M : M + 1)

Description: Compares the contents of index register X with a 16-bit value at the address specified and sets the condition codes accordingly. The compare is accomplished internally by doing a 16-bit subtract of (M : M + 1) from index register X without modifying either index register X or (M : M + 1).

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \overline{R12} \cdot \overline{R11} \cdot \overline{R10} \cdot \overline{R9} \cdot \overline{R8} \cdot \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
Set if result is $0000; cleared otherwise.

V $IX15 \cdot \overline{M15} \cdot \overline{R15} + \overline{IX15} \cdot M15 \cdot R15$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{IX15} \cdot M15 + M15 \cdot R15 + R15 \cdot \overline{IX15}$
Set if the absolute value of the contents of memory is larger than the absolute value of the index register; cleared otherwise.

Source Form: CPX (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	CPX (IMM)			CPX (DIR)			CPX (EXT)			CPX (IND,X)			CPX (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	8C	1	OP	9C	1	OP	BC	1	OP	AC	1	OP	CD	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AC	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	FFFF	—	1	00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5				FFFF	—	1	hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6							FFFF	—	1	FFFF	—	1	Y + ff + 1	(Y + ff + 1)	1
7													FFFF	—	1

DAA — Decimal Adjust Accumulator A — DAA

Operation: The following table summarizes the operation of the DAA instruction for all legal combinations of input operands. A correction factor (column 5 in the following table) is added to ACCA to restore the result of an addition of two BCD operands to a valid BCD value and set or clear the carry bit.

State of C Bit Before DAA (Column 1)	Upper Half-Byte of ACCA (Bits [7:4]) (Column 2)	Initial Half-Carry H Bit from CCR (Column 3)	Lower Half-Byte of ACCA (Bits [3:0]) (Column 4)	Number Added to ACCA by DAA (Column 5)	State of C Bit After DAA (Column 6)
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	0-3	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	1	0-3	66	1
1	0-2	0	0-9	60	1
1	0-2	0	A-F	66	1
1	0-3	1	0-3	66	1

NOTE: *Columns (1) through (4) of the above table represent all possible cases which can result from any of the operations ABA, ADD, or ADC, with initial carry either set or clear, applied to two binary-coded-decimal operands. The table shows hexadecimal values.*

Description: If the contents of ACCA and the state of the carry/borrow bit C and the state of the half-carry bit H are all the result of applying any of the operations ABA, ADD, or ADC to binary-coded-decimal operands, with or without an initial carry, the DAA operation will adjust the contents of ACCA and the carry bit C in the CCR to represent the correct binary-coded-decimal sum and the correct state of the C bit.

Decimal Adjust Accumulator A (Continued)

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	?	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V ?
Not defined

C See table above

Source Form: DAA

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	DAA (INH)		
	Addr	Data	R/W
1	OP	19	1
2	OP + 1	—	1

For the purpose of illustration, consider the case where the BCD value $99 was just added to the BCD value $22. The add instruction is a binary operation, which yields the result $BB with no carry (C) or half carry (H). This corresponds to the fifth row of the table on the previous page. The DAA instruction, therefore, will add the correction factor $66 to the result of the addition, giving a result of $21 with the carry bit set. This result corresponds to the BCD value $121, which is the expected BCD result.

DEC — Decrement — DEC

Operation: ACCX ⇐ (ACCX) – $01 **or:** M ⇐ (M) – $01

Description: Subtract one from the contents of ACCX or M.

The N, Z, and V bits in the CCR are set or cleared according to the results of the operation. The C bit in the CCR is not affected by the operation, thus allowing the DEC instruction to be used as a loop counter in multiple-precision computations.

When operating on unsigned values, only BEQ and BNE branches can be expected to perform consistently. When operating on twos complement values, all signed branches are available.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $X7 \bullet \overline{X6} \bullet \overline{X5} \bullet \overline{X4} \bullet X3 \bullet \overline{X2} \bullet \overline{X1} \bullet \overline{X0} = \overline{R7} \bullet R6 \bullet R5 \bullet R4 \bullet R3 \bullet R2 \bullet R1 \bullet R0$
Set if a twos complement overflow resulted from the operation; cleared otherwise. Twos complement overflow occurs if and only if (ACCX) or (M) was $80 before the operation.

Source Form: DECA; DECB; DEC (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	DECA (INH)			DECB (INH)			DEC (EXT)			DEC (IND,X)			DEC (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	4A	1	OP	5A	1	OP	7A	1	OP	6A	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	6A	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	1

DES — Decrement Stack Pointer — DES

Operation: SP ⇐ (SP) – $0001

Description: Subtract one from the stack pointer.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: DES

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	DES (INH)		
	Addr	Data	R/W
1	OP	34	1
2	OP + 1	—	1
3	SP	—	1

DEX — Decrement Index Register X — DEX

Operation: IX ⇐ (IX) – $0001

Description: Subtract one from index register X

Only the Z bit is set or cleared according to the result of this operation.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	⇕	—	—

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

Source Form: DEX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	DEX (INH)		
	Addr	Data	R/W
1	OP	09	1
2	OP + 1	—	1
3	FFFF	—	1

EOR — Exclusive OR — EOR

Operation: ACCX ⇐ (ACCX) ⊕ (M)

Description: Performs the logical exclusive-OR between the contents of ACCX and the contents of M and places the result in ACCX. (Each bit of ACCX after the operation will be the logical exclusive-OR of the corresponding bits of M and ACCX before the operation.)

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇳	⇳	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

Source Forms: EORA (opr); EORB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	EORA (IMM)			EORA (DIR)			EORA (EXT)			EORA (IND,X)			EORA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	88	1	OP	98	1	OP	B8	1	OP	A8	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A8	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	EORB (IMM)			EORB (DIR)			EORB (EXT)			EORB (IND,X)			EORB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C8	1	OP	D8	1	OP	F8	1	OP	E8	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E8	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

INC — Increment — INC

Operation: ACCX ⇐ (ACCX) + $01 **or:** M ⇐ (M) + $01

Description: Add one to the contents of ACCX or M.

The N, Z, and V bits in the CCR are set or cleared according to the results of the operation. The C bit in the CCR is not affected by the operation, thus allowing the INC instruction to be used as a loop counter in multiple-precision computations.

When operating on unsigned values, only BEQ and BNE branches can be expected to perform consistently. When operating on twos-complement values, all signed branches are available.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	—

N R7
Set is MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $\overline{X7} \bullet X6 \bullet X5 \bullet X4 \bullet X3 \bullet X2 \bullet X1 \bullet X0$
Set if there is a twos complement overflow as a result of the operation; cleared otherwise. Two's complement overflow occurs if and only if (ACCX) or (M) was $7F before the operation.

Source Forms: INCA; INCB; INC (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	INCA (INH)			INCB (INH)			INC (EXT)			INC (IND,X)			INC (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	4C	1	OP	5C	1	OP	7C	1	OP	6C	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	6C	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

INS — Increment Stack Pointer — INS

Operation: SP ⇐ (SP) + $0001

Description: Adds one to the stack pointer

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: INS

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	INS (INH)		
	Addr	Data	R/W
1	OP	31	1
2	OP + 1	—	1
3	SP	—	1

INX — Increment Index Register X — INX

Operation: IX ⇐ (IX) + $0001

Description: Adds one to index register X

Only the Z bit is set or cleared according to the result of this operation.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	⇳	—	—

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

Source Form: INX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	INX (INH)		
	Addr	Data	R/W
1	OP	08	1
2	OP + 1	—	1
3	FFFF	—	1

JMP — Jump — JMP

Operation: PC ⇐ Effective Address

Description: A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended or INDexed addressing.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: JMP (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	JMP (EXT)			JMP (IND,X)			JMP (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	7E	1	OP	6E	1	OP	18	1
2	OP + 1	hh	1	OP + 1	ff	1	OP + 1	6E	1
3	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							FFFF	—	1

JSR — Jump to Subroutine — JSR

Operation:

PC ⇐ (PC) + $0003 (for EXTended or INDexed, Y addressing)
or:
PC ⇐ (PC) + $0002 (for DIRect or INDexed, X addressing)
⇓ (PCL) Push low-order return address onto stack
SP ⇐ (SP) – $0001
⇓ (PCH) Push high-order return address onto stack
SP ⇐ (SP) – $0001
PC ⇐ Effective Addr Load start address or requested subroutine

Description:

The program counter is incremented by three or by two, depending on the addressing mode, and is then pushed onto the stack, eight bits at a time, least significant byte first. The stack pointer points to the next empty location in the stack. A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended, DIRect, or INDexed addressing.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: JSR (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	JSR (DIR)			JSR (EXT)			JSR (IND,X)			JSR (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	9D	1	OP	BD	1	OP	AD	1	OP	18	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AD	1
3	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	SP	Rtn lo	0	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5	SP – 1	Rtn hi	0	SP	Rtn lo	0	SP	Rtn lo	0	Y + ff	(Y + ff)	1
6				SP – 1	Rtn hi	0	SP – 1	Rtn hi	0	SP	Rtn lo	0
7										SP – 1	Rtn hi	0

LDA — Load Accumulator — LDA

Operation: ACCX ⇐ (M)

Description: Loads the contents of memory into the 8-bit accumulator. The condition codes are set according to the data.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

Source Forms: LDAA (opr); LDAB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LDAA (IMM)			LDAA (DIR)			LDAA (EXT)			LDAA (IND,X)			LDAA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	86	1	OP	96	1	OP	B6	1	OP	A6	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A6	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	LDAB (IMM)			LDAB (DIR)			LDAB (EXT)			LDAB (IND,X)			LDAB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C6	1	OP	D6	1	OP	F6	1	OP	E6	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E6	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

LDD — Load Double Accumulator — LDD

Operation: ACCX ⇐ (M : M + 1); ACCA ⇐ (M), ACCB ⇐ (M + 1)

Description: Loads the contents of memory locations M and M + 1 into the double accumulator D. The condition codes are set according to the data. The information from location M is loaded into accumulator A, and the information from location M + 1 is loaded into accumulator B.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: LDD (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LDD (IMM)			LDD (DIR)			LDD (EXT)			LDD (IND,X)			LDD (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	CC	1	OP	DC	1	OP	FC	1	OP	EC	1	OP	18	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EC	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4				00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6													Y + ff + 1	(Y + ff + 1)	1

LDS — Load Stack Pointer — LDS

Operation: SPH ⇐ (M), SPL ⇐ (M + 1)

Description: Loads the most significant byte of the stack pointer from the byte of memory at the address specified by the program, and loads the least significant byte of the stack pointer from the next byte of memory at one plus the address specified by the program.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: LDS (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LDS (IMM)			LDS (DIR)			LDS (EXT)			LDS (IND,X)			LDS (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	8E	1	OP	9E	1	OP	BE	1	OP	AE	1	OP	18	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AE	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4				00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6													Y + ff + 1	(Y + ff + 1)	1

LDX — Load Index Register X — LDX

Operation: IXH ⇐ (M), IXL ⇐ (M + 1)

Description: Loads the most significant byte of index register X from the byte of memory at the address specified by the program, and loads the least significant byte of index register X from the next byte of memory at one plus the address specified by the program.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \overline{R12} \cdot \overline{R11} \cdot \overline{R10} \cdot \overline{R9} \cdot \overline{R8} \cdot \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: LDX (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

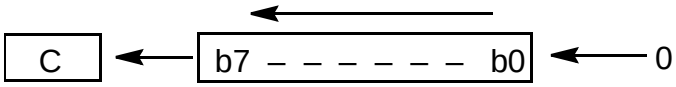
Cycle	LDX (IMM)			LDX (DIR)			LDX (EXT)			LDX (IND,X)			LDX (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	CE	1	OP	DE	1	OP	FE	1	OP	EE	1	OP	CD	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EE	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4				00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6													Y + ff + 1	(Y + ff + 1)	1

LSL

Logical Shift Left
(Same as ASL)

LSL

Operation:



Description: Shifts all bits of the ACCX or M one place to the left. Bit 0 is loaded with a 0. The C bit in the CCR is loaded from the most significant bit of ACCX or M.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the shift).

C M7
Set if, before the shift, the MSB of ACCX or M was set; cleared otherwise.

Source Forms: LSLA; LSLB; LSL (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

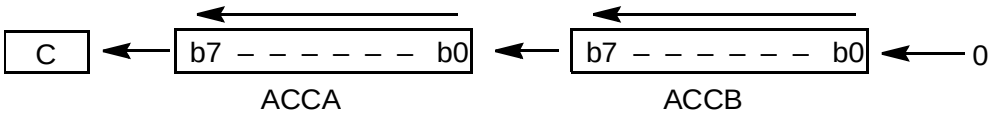
Cycle	LSLA (INH)			LSLB (INH)			LSL (EXT)			LSL (IND,X)			LSL (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	48	1	OP	58	1	OP	78	1	OP	68	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	68	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

LSLD

Logical Shift Left Double (Same as ASLD)

LSLD

Operation:



Description: Shifts all bits of ACCD one place to the left. Bit 0 is loaded with a 0. The C bit in the CCR is loaded from the most significant bit of ACCD.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the shift).

C D15
Set if, before the shift, the MSB of ACCD was set; cleared otherwise.

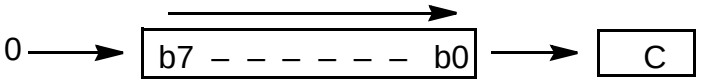
Source Form: LSLD

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LSLD (INH)		
	Addr	Data	R/W
1	OP	05	1
2	OP + 1	—	1
3	FFFF	—	1

LSR — Logical Shift Right — LSR

Operation:



Description: Shifts all bits of the ACCX or M one place to the right. Bit 7 is loaded with 0. The C bit is loaded from the least significant bit of ACCX or M.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	0	⇕	⇕	⇕

N 0
Cleared.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the shift)
Since N = 0, this simplifies to C (after the shift).

C M0
Set if, before the shift, the LSB of ACCX or M was set; cleared otherwise.

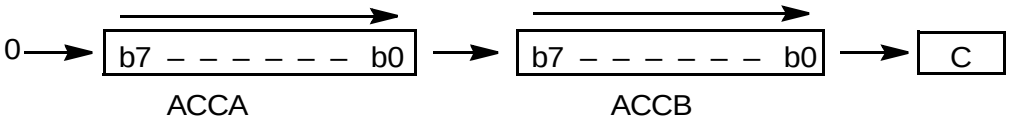
Source Forms: LSRA; LSRB; LSR (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LSRA (INH)			LSRB (INH)			LSR (EXT)			LSR (IND,X)			LSR (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	44	1	OP	54	1	OP	74	1	OP	64	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	64	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

LSRD Logical Shift Right Double Accumulator LSRD

Operation:



Description: Shifts all bits of ACCD one place to the right. Bit 15 (MSB of ACCA) is loaded with 0. The C bit is loaded from the least significant bit of ACCD (LSB of ACCB).

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	0	⇕	⇕	⇕

N 0
Cleared.

Z $\overline{R15} \cdot \overline{R14} \cdot \overline{R13} \cdot \overline{R12} \cdot \overline{R11} \cdot \overline{R10} \cdot \overline{R9} \cdot \overline{R8} \cdot \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
Set if result is $0000; cleared otherwise.

V D0
Set if, after the shift operation, C is set; cleared otherwise.

C D0
Set if, before the shift, the least significant bit of ACCD was set; cleared otherwise.

Source Form: LSRD

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	LSRD (INH)		
	Addr	Data	R/W
1	OP	04	1
2	OP + 1	—	1
3	FFFF	—	1

MUL — Multiply Unsigned — MUL

Operation: ACCD ⇐ (ACCA) × (ACCB)

Description: Multiplies the 8-bit unsigned binary value in accumulator A by the 8-bit unsigned binary value in accumulator B to obtain a 16-bit unsigned result in the double accumulator D. Unsigned multiply allows multiple-precision operations. The carry flag allows rounding the most significant byte of the result through the sequence MUL, ADCA #0.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	⇕

C R7
Set if bit 7 of the result (ACCB bit 7) is set; cleared otherwise.

Source Form: MUL

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	MUL (INH)		
	Addr	Data	R/W
1	OP	3D	1
2	OP + 1	—	1
3–10	FFFF	—	1

NEG — Negate — NEG

Operation: ACCX ⇐ – (ACCX) = $00 – (ACCX) **or:** M ⇐ – (M) = $00 – (M)

Description: Replaces the contents of ACCX or M with its twos complement; the value $80 is left unchanged

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $R7 \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if there is a twos complement overflow from the implied subtraction from zero; cleared otherwise. A twos complement overflow will occur if and only if the contents of ACCX or M is $80.

C R7 + R6 + R5 + R4 + R3 + R2 + R1 + R0
Set if there is a borrow in the implied subtraction from zero; cleared otherwise. The C bit will be set in all cases except when the contents of ACCX or M is $00.

Source Forms: NEGA; NEGB; NEG (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	NEGA (INH)			NEGB (INH)			NEG (EXT)			NEG (IND,X)			NEG (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	40	1	OP	50	1	OP	70	1	OP	60	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	60	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

NOP — No Operation — NOP

Description: This is a single-byte instruction that causes only the program counter to be incremented. No other registers are affected. This instruction is typically used to produce a time delay although some software disciplines discourage CPU frequency-based time delays. During debug, NOP instructions are sometimes used to temporarily replace other machine code instructions, thus disabling the replaced instructions.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: NOP

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	NOP (INH)		
	Addr	Data	R/W
1	OP	01	1
2	OP + 1	—	1

ORA — Inclusive OR — ORA

Operation: ACCX ⇐ (ACCX) + (M)

Description: Performs the logical inclusive-OR between the contents of ACCX and the contents of M and places the result in ACCX. (Each bit of ACCX after the operation will be the logical inclusive-OR of the corresponding bits of M and ACCX before the operation.)

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇳	⇳	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

Source Forms: ORAA (opr); ORAB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ORAA (IMM)			ORAA (DIR)			ORAA (EXT)			ORAA (IND,X)			ORAA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	8A	1	OP	9A	1	OP	BA	1	OP	AA	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AA	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	ORAB (IMM)			ORAB (DIR)			ORAB (EXT)			ORAB (IND,X)			ORAB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	CA	1	OP	DA	1	OP	FA	1	OP	EA	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EA	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

PSH — Push Data onto Stack — PSH

Operation: ⇓ ACCX, SP ⇐ (SP) – $0001

Description: The contents of ACCX are stored on the stack at the address contained in the stack pointer. The stack pointer is then decremented.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Forms: PSHA; PSHB

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PSHA (INH)			PSHB (INH)		
	Addr	Data	R/W	Addr	Data	R/W
1	OP	36	1	OP	37	1
2	OP + 1	—	1	OP + 1	—	1
3	SP	(A)	0	SP	(B)	0

PSHX — Push Index Register X onto Stack — PSHX

Operation: ⇓ (IXL), SP ⇐ (SP) – $0001
⇓ (IXH), SP ⇐ (SP) – $0001

Description: The contents of index register X are pushed onto the stack (low-order byte first) at the address contained in the stack pointer. The stack pointer is then decremented by two.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: PSHX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PSHX (INH)		
	Addr	Data	R/W
1	OP	3C	1
2	OP + 1	—	1
3	SP	(IXL)	0
4	SP – 1	(IXH)	0

PUL — Pull Data from Stack — PUL

Operation: SP ⇐ (SP) + $0001, ⇑ (ACCX)

Description: The stack pointer is incremented. The ACCX is then loaded from the stack at the address contained in the stack pointer.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Forms: PULA; PULB

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PULA (INH)			PULB (INH)		
	Addr	Data	R/W	Addr	Data	R/W
1	OP	32	1	OP	33	1
2	OP + 1	—	1	OP + 1	—	1
3	SP	—	1	SP	—	1
4	SP + 1	get A	1	SP + 1	get B	1

PULX — Pull Index Register X from Stack — PULX

Operation: SP ⇐ (SP) + $0001; ⇑ (IXH)
SP ⇐ (SP) + $0001; ⇑ (IXL)

Description: Index register X is pulled from the stack (high-order byte first) beginning at the address contained in the stack pointer plus one. The stack pointer is incremented by two in total.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

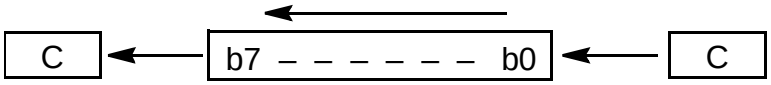
Source Form: PULX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PULX (INH)		
	Addr	Data	R/W
1	OP	38	1
2	OP + 1	—	1
3	SP	—	1
4	SP + 1	get IXH	1
5	SP + 2	get IXL	1

ROL — Rotate Left — ROL

Operation:



Description: Shifts all bits of the ACCX or M one place to the left. Bit 0 is loaded from the C bit. The C bit in the CCR is loaded from the most significant bit of ACCX or M. The rotate operations include the carry bit to allow extension of the shift and rotate operations to multiple bytes. For example, to shift a 24-bit value left one bit, the sequence ASL LOW, ROL MID, ROL HIGH could be used where LOW, MID, and HIGH refer to the low-order, middle, and high-order bytes of the 24-bit value, respectively.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the rotate)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the rotate).

C M7
Set if, before the rotate, the MSB of ACCX or M was set; cleared otherwise.

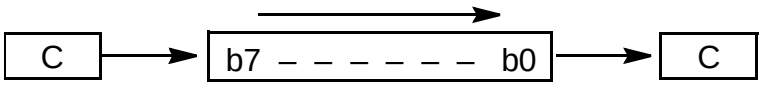
Source Forms: ROLA; ROLB; ROL (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	ROLA (INH)			ROLB (INH)			ROL (EXT)			ROL (IND,X)			ROL (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	49	1	OP	59	1	OP	79	1	OP	69	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	69	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

ROR — Rotate Right — ROR

Operation:



Description: Shifts all bits of the ACCX or M one place to the right. Bit 7 is loaded from the C bit. The C bit in the CCR is loaded from the least significant bit of ACCX or M. The rotate operations include the carry bit to allow extension of the shift and rotate operations to multiple bytes. For example, to shift a 24-bit value right one bit, the sequence LSR HIGH, ROR MID, ROR LOW could be used where LOW, MID, and HIGH refer to the low-order, middle, and high-order bytes of the 24-bit value, respectively. The first LSR could be replaced by ASR to maintain the original value of the sign bit (MSB of high-order byte) of the 24-bit value.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇳	⇳	⇳	⇳

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $N \oplus C = [N \bullet \overline{C}] + [\overline{N} \bullet C]$ (for N and C after the rotate)
Set if (N is set and C is clear) or (N is clear and C is set); cleared otherwise (for values of N and C after the rotate).

C M0
Set if, before the rotate, the LSB of ACCX or M was set; cleared otherwise.

Source Forms: RORA; RORB; ROR (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	RORA (INH)			RORB (INH)			ROR (EXT)			ROR (IND,X)			ROR (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	46	1	OP	56	1	OP	76	1	OP	66	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	66	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							hhll	result	0	X + ff	result	0	FFFF	—	1
7													Y + ff	result	0

RTI — Return from Interrupt — RTI

Operation:

SP ⇐ (SP) + $0001, ⇑ (CCR)
SP ⇐ (SP) + $0001, ⇑ (ACCB)
SP ⇐ (SP) + $0001, ⇑ (ACCA)
SP ⇐ (SP) + $0001, ⇑ (IXH)
SP ⇐ (SP) + $0001, ⇑ (IXL)
SP ⇐ (SP) + $0001, ⇑ (IYH)
SP ⇐ (SP) + $0001, ⇑ (IYL)
SP ⇐ (SP) + $0001, ⇑ (PCH)
SP ⇐ (SP) + $0001, ⇑ (PCL)

Description:

The condition code, accumulators B and A, index registers X and Y, and the program counter will be restored to a state pulled from the stack. The X bit in the CCR may be cleared as a result of an RTI instruction but may not be set if it was cleared prior to execution of the RTI instruction.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
⇕	⇓	⇕	⇕	⇕	⇕	⇕	⇕

Condition code bits take on the value of the corresponding bit of the unstacked CCR except that the X bit may not change from a 0 to a 1. Software can leave X set, leave X clear, or change X from 1 to 0.
The XIRQ interrupt mask can become set only as a result of a reset or recognition of an $\overline{\text{XIRQ}}$ interrupt.

Source Form: RTI

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	RTI (INH)		
	Addr	Data	R/W
1	OP	3B	1
2	OP + 1	—	1
3	SP	—	1
4	SP + 1	get CC	1
5	SP + 2	get B	1
6	SP + 3	get A	1
7	SP + 4	get IXH	1
8	SP + 5	get IXL	1
9	SP + 6	get IYH	1
10	SP + 7	get IYL	1
11	SP + 8	Rtn hi	1
12	SP + 9	Rtn lo	1

RTS — Return from Subroutine — RTS

Operation: SP ⇐ (SP) + $0001, ⇑ (PCH)
SP ⇐ (SP) + $0001, ⇑ (PCL)

Description: The stack pointer is incremented by one. The contents of the byte of memory, at the address now contained in the stack pointer, are loaded into the high-order eight bits of the program counter. The stack pointer is again incremented by one. The contents of the byte of memory, at the address now contained in the stack pointer, are loaded into the low-order eight bits of the program counter.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: RTS

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	RTS (INH)		
	Addr	Data	R/W
1	OP	39	1
2	OP + 1	—	1
3	SP	—	1
4	SP + 1	Rtn hi	1
5	SP + 2	Rtn lo	1

SBA — Subtract Accumulators — SBA

Operation: ACCA ⇐ (ACCA) – (ACCB)

Description: Subtracts the contents of ACCB from the contents of ACCA and places the result in ACCA. The contents of ACCB are not affected. For subtract instructions, the C bit in the CCR represents a borrow.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $A7 \bullet \overline{B7} \bullet \overline{R7} + \overline{A7} \bullet B7 \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{A7} \bullet B7 + B7 \bullet R7 + R7 \bullet \overline{A7}$
Set if the absolute value of ACCB is larger than the absolute value of ACCA; cleared otherwise.

Source Form: SBA

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SBA (INH)		
	Addr	Data	R/W
1	OP	10	1
2	OP + 1	—	1

SBC — Subtract with Carry — SBC

Operation: ACCX ⇐ (ACCX) – (M) – (C)

Description: Subtracts the contents of M and the contents of C from the contents of ACCX and places the result in ACCX. For subtract instructions, the C bit in the CCR represents a borrow.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $X7 \bullet \overline{M7} \bullet \overline{R7} + \overline{X7} \bullet M7 \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{X7} \bullet M7 + M7 \bullet R7 + R7 \bullet \overline{X7}$
Set if the absolute value of the contents of memory plus previous carry is larger than the absolute value of the accumulator; cleared otherwise.

Source Forms: SBCA (opr); SBCB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SBCA (IMM)			SBCA (DIR)			SBCA (EXT)			SBCA (IND,X)			SBCA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	82	1	OP	92	1	OP	B2	1	OP	A2	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A2	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	SBCB (IMM)			SBCB (DIR)			SBCB (EXT)			SBCB (IND,X)			SBCB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C2	1	OP	D2	1	OP	F2	1	OP	E2	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E2	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

SEC — Set Carry — SEC

Operation: C bit ⇐ 1

Description: Sets the C bit in the CCR.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	1

C 1
Set

Source Form: SEC

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SEC (INH)		
	Addr	Data	R/W
1	OP	0D	1
2	OP + 1	—	1

SEI — Set Interrupt Mask — SEI

Operation: I bit ⇐ 1

Description: Sets the interrupt mask bit in the CCR. When the I bet is set, all maskable interrupts are inhibited, and the MPU will recognize only non-maskable interrupt sources or an SWI.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	1	—	—	—	—

I 1
Set

Source Form: SEI

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SEI (INH)		
	Addr	Data	R/W
1	OP	0F	1
2	OP + 1	—	1

SEV — Set Two's Complement Overflow Bit — SEV

Operation: V bit ⇐ 1

Description: Sets the twos complement overflow bit in the CCR.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	1	—

V 1
Set

Source Form: SEV

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SEV (INH)		
	Addr	Data	R/W
1	OP	0B	1
2	OP + 1	—	1

STA — Store Accumulator — STA

Operation: M ⇐ (ACCX)

Description: Stores the contents of ACCX in memory. The contents of ACCX remain the same.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N X7
Set if MSB of result is set; cleared otherwise.

Z $\overline{X7} \bullet \overline{X6} \bullet \overline{X5} \bullet \overline{X4} \bullet \overline{X3} \bullet \overline{X2} \bullet \overline{X1} \bullet \overline{X0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

Source Forms: STAA (opr); STAB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	STAA (DIR)			STAA (EXT)			STAA (IND,X)			STAA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	97	1	OP	B7	1	OP	A7	1	OP	18	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A7	1
3	00dd	(A)	0	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4				hhll	(A)	0	X + ff	(A)	0	FFFF	—	1
5										Y + ff	(A)	0

Cycle	STAB (DIR)			STAB (EXT)			STAB (IND,X)			STAB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	D7	1	OP	F7	1	OP	E7	1	OP	18	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E7	1
3	00dd	(B)	0	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4				hhll	(B)	0	X + ff	(B)	0	FFFF	—	1
5										Y + ff	(B)	0

STD — Store Double Accumulator — STD

Operation: M : M + 1 ⇐ (ACCD); M ⇐ (ACCA), M + 1 ⇐ (ACCB)

Description: Stores the contents of double accumulator ACCD in memory. The contents of ACCD remain unchanged.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N D15
Set if MSB of result is set; cleared otherwise.

Z $\overline{D15} \bullet \overline{D14} \bullet \overline{D13} \bullet \overline{D12} \bullet \overline{D11} \bullet \overline{D10} \bullet \overline{D9} \bullet \overline{D8} \bullet \overline{D7} \bullet \overline{D6} \bullet \overline{D5} \bullet \overline{D4} \bullet \overline{D3} \bullet \overline{D2} \bullet \overline{D1} \bullet \overline{D0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: STD (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	STD (DIR)			STD (EXT)			STD (IND,X)			STD (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	DD	1	OP	FD	1	OP	ED	1	OP	18	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	ED	1
3	00dd	(A)	0	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	00dd + 1	(B)	0	hhll	(A)	0	X + ff	(A)	0	FFFF	—	1
5				hhll + 1	(B)	0	X + ff + 1	(B)	0	Y + ff	(A)	0
6										Y + ff + 1	(B)	0

STS — Store Stack Pointer — STS

Operation: M ⇐ (SPH), M + 1 ⇐ (SPL)

Description: Stores the most significant byte of the stack pointer in memory at the address specified by the program and stores the least significant byte of the stack pointer at the next location in memory, at one plus the address specified by the program.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N SP15
Set if MSB of result is set; cleared otherwise.

Z $\overline{SP15} \cdot \overline{SP14} \cdot \overline{SP13} \cdot \overline{SP12} \cdot \overline{SP11} \cdot \overline{SP10} \cdot \overline{SP9} \cdot \overline{SP8} \cdot \overline{SP7} \cdot \overline{SP6} \cdot \overline{SP5} \cdot \overline{SP4} \cdot \overline{SP3} \cdot \overline{SP2} \cdot \overline{SP1} \cdot \overline{SP0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: STS (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	STS (DIR)			STS (EXT)			STS (IND,X)			STS (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	9F	1	OP	BF	1	OP	AF	1	OP	18	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AF	1
3	00dd	(SPH)	0	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	00dd + 1	(SPL)	0	hhll	(SPH)	0	X + ff	(SPH)	0	FFFF	—	1
5				hhll + 1	(SPL)	0	X + ff + 1	(SPL)	0	Y + ff	(SPH)	0
6										Y + ff + 1	(SPL)	0

STX — Store Index Register X — STX

Operation: M ⇐ (IXH), M + 1 ⇐ (IXL)

Description: Stores the most significant byte of index register X in memory at the address specified by the program, and stores the least significant byte of index register X at the next location in memory, at one plus the address specified by the program.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N IX15
Set if MSB of result is set; cleared otherwise.

Z $\overline{IX15} \bullet \overline{IX14} \bullet \overline{IX13} \bullet \overline{IX12} \bullet \overline{IX11} \bullet \overline{IX10} \bullet \overline{IX9} \bullet \overline{IX8} \bullet \overline{IX7} \bullet \overline{IX6} \bullet \overline{IX5} \bullet \overline{IX4} \bullet \overline{IX3} \bullet \overline{IX2} \bullet \overline{IX1} \bullet \overline{IX0}$
Set if result is $0000; cleared otherwise.

V 0
Cleared

Source Form: STX (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	STX (DIR)			STX (EXT)			STX (IND,X)			STX (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	DF	1	OP	FF	1	OP	EF	1	OP	CD	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	EF	1
3	00dd	(IXH)	0	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	00dd + 1	(IXL)	0	hhll	(IXH)	0	X + ff	(IXH)	0	FFFF	—	1
5				hhll + 1	(IXL)	0	X + ff + 1	(IXL)	0	Y + ff	(IXH)	0
6										Y + ff + 1	(IXL)	0

SUB — Subtract — SUB

Operation: ACCX ⇐ (ACCX) – (M)

Description: Subtracts the contents of M from the contents of ACCX and places the result in ACCX. For subtract instructions, the C bit in the CCR represents a borrow.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V $X7 \bullet \overline{M7} \bullet \overline{R7} + \overline{X7} \bullet M7 \bullet R7$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{X7} \bullet M7 + M7 \bullet R7 + R7 \bullet \overline{X7}$
Set if the absolute value of the contents of memory plus previous carry is larger than the absolute value of the accumulator; cleared otherwise.

Source Forms: SUBA (opr); SUBB (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SUBA (IMM)			SUBA (DIR)			SUBA (EXT)			SUBA (IND,X)			SUBA (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	80	1	OP	90	1	OP	B0	1	OP	A0	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A0	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

Cycle	SUBB (IMM)			SUBB (DIR)			SUBB (EXT)			SUBB (IND,X)			SUBB (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	C0	1	OP	D0	1	OP	F0	1	OP	E0	1	OP	18	1
2	OP + 1	ii	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	E0	1
3				00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5													Y + ff	(Y + ff)	1

SUBD — Subtract Double Accumulator — SUBD

Operation: ACCD ⇐ (ACCD) – (M : M + 1)

Description: Subtracts the contents of M : M + 1 from the contents of double accumulator D and places the result in ACCD. For subtract instructions, the C bit in the CCR represents a borrow.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	⇕	⇕

N R15
Set if MSB of result is set; cleared otherwise.

Z $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $0000; cleared otherwise.

V $D15 \bullet \overline{M15} \bullet \overline{R15} + \overline{D15} \bullet M15 \bullet R15$
Set if a twos complement overflow resulted from the operation; cleared otherwise.

C $\overline{D15} \bullet M15 + M15 \bullet R15 + R15 \bullet \overline{D15}$
Set if the absolute value of the contents of memory is larger than the absolute value of the accumulator; cleared otherwise.

Source Form: SUBD (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SUBD (IMM)			SUBD (DIR)			SUBD (EXT)			SUBD (IND,X)			SUBD (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	83	1	OP	93	1	OP	B3	1	OP	A3	1	OP	18	1
2	OP + 1	jj	1	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	A3	1
3	OP + 2	kk	1	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	FFFF	—	1	00dd + 1	(00dd + 1)	1	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5				FFFF	—	1	hhll + 1	(hhll + 1)	1	X + ff + 1	(X + ff + 1)	1	Y + ff	(Y + ff)	1
6							FFFF	—	1	FFFF	—	1	Y + ff + 1	(Y + ff + 1)	1
7													FFFF	—	1

SWI — Software Interrupt — SWI

Operation:

PC ⇐ (PC) + $0001
⇓ (PCL), SP ⇐ (SP) – $0001
⇓ (PCH), SP ⇐ (SP) – $0001
⇓ (IYL), SP ⇐ (SP) – $0001
⇓ (IYH), SP ⇐ (SP) – $0001
⇓ (IXL), SP ⇐ (SP) – $0001
⇓ (IXH), SP ⇐ (SP) – $0001
⇓ (ACCA), SP ⇐ (SP) – $0001
⇓ (ACCB), SP ⇐ (SP) – $0001
⇓ (CCR), SP ⇐ (SP) – $0001
I ⇐ 1, PC ⇐ (SWI vector)

Description:

The program counter is incremented by one. The program counter, index registers Y and X, and accumulators A and B are pushed onto the stack. The CCR is then pushed onto the stack. The stack pointer is decremented by one after each byte of data is stored on the stack. The I bit in the CCR is then set. The program counter is loaded with the address stored at the SWI vector, and instruction execution resumes at this location. This instruction is not maskable by the I bit.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	1	—	—	—	—

I 1
Set

Source Form: SWI

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	SWI (INH)		
	Addr	Data	R/W
1	OP	3F	1
2	OP + 1	—	1
3	SP	Rtn lo	0
4	SP – 1	Rtn hi	0
5	SP – 2	(IYL)	0
6	SP – 3	(IYH)	0
7	SP – 4	(IXL)	0
8	SP – 5	(IXH)	0
9	SP – 6	(A)	0
10	SP – 7	(B)	0
11	SP – 8	(CCR)	0
12	SP – 8	(CCR)	1
13	Vec hi	Svc hi	1
14	Vec lo	Svc lo	1

TAB — Transfer from Accumulator A to B — TAB

Operation: ACCB ⇐ (ACCA)

Description: Moves the contents of ACCA to ACCB. The former contents of ACCB are lost; the contents of ACCA are not affected.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared.

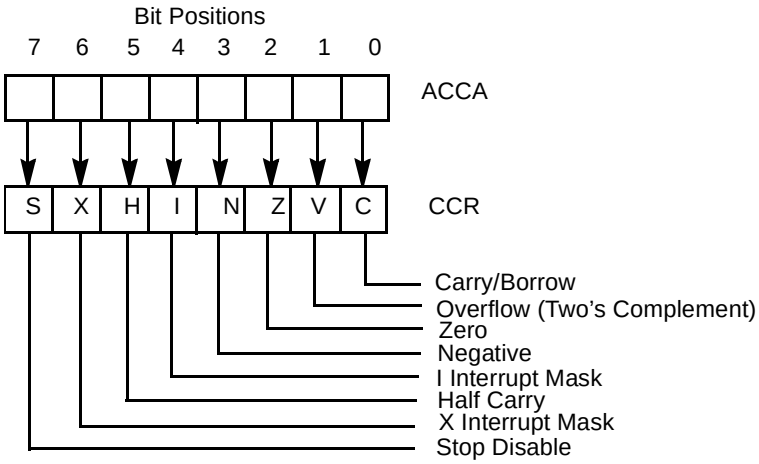
Source Form: TAB

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TAB (INH)		
	Addr	Data	R/W
1	OP	16	1
2	OP + 1	—	1

TAP — Transfer from Accumulator A to CCR — TAP

Operation: CCR ⇐ (ACCA)



Description: Transfers the contents of bit positions 7–0 of accumulator A to the corresponding bit positions of the CCR. The contents of accumulator A remain unchanged. The X bit in the CCR may be cleared as a result of a TAP instruction but may not be set if it was clear prior to execution of the TAP instruction.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
⇳	⇓	⇳	⇳	⇳	⇳	⇳	⇳

Condition code bits take on the value of the corresponding bit of accumulator A except that the X bit may not change from a 0 to a 1. Software can leave X set, leave X clear, or change X from 1 to 0. The XIRQ interrupt mask can become set only as a result of a reset or recognition of an $\overline{\text{XIRQ}}$ interrupt.

Source Form: TAP

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TAP (INH)		
	Addr	Data	R/W
1	OP	06	1
2	OP + 1	—	1

TBA — Transfer from Accumulator B to A — TBA

Operation: ACCA ⇐ (ACCB)

Description: Moves the contents of ACCB to ACCA. The former contents of ACCA are lost; the contents of ACCB are not affected.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	—

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if result is $00; cleared otherwise.

V 0
Cleared.

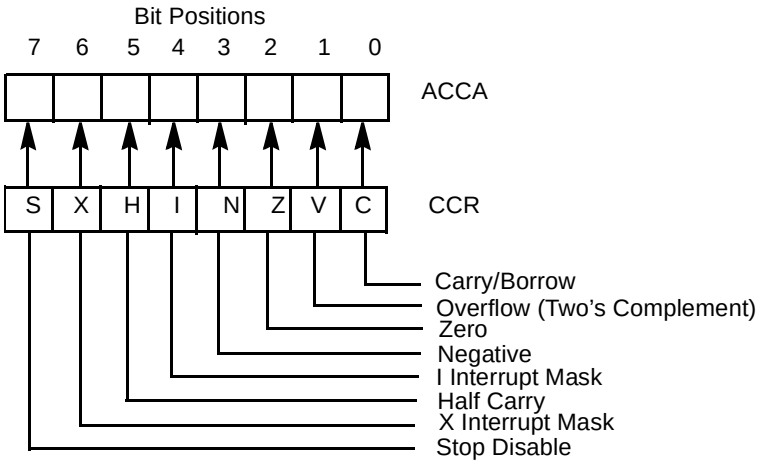
Source Form: TBA

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TBA (INH)		
	Addr	Data	R/W
1	OP	17	1
2	OP + 1	—	1

TPA Transfer from CCR to Accumulator A TPA

Operation: ACCA ⇐ (CCR)



Description: Transfers the contents of the CCR to corresponding bit positions of accumulator A. The CCR remains unchanged.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: TPA

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TPA (INH)		
	Addr	Data	R/W
1	OP	07	1
2	OP + 1	—	1

TST — Test — TST

Operation: (ACCX) – $00 **or:** (M) – $00

Description: Subtracts $00 from the contents of ACCX or M and sets the condition codes accordingly.

The subtraction is accomplished internally without modifying either ACCX or M.

The TST instruction provides only minimum information when testing unsigned values. Since no unsigned value is less than zero, BLO and BLS have no utility. While BHI could be used after TST, it provides exactly the same control as BNE, which is preferred. After testing signed values, all signed branches are available.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	⇕	⇕	0	0

N R7
Set if MSB of result is set; cleared otherwise.

Z $\overline{M7} \bullet \overline{M6} \bullet \overline{M5} \bullet \overline{M4} \bullet \overline{M3} \bullet \overline{M2} \bullet \overline{M1} \bullet \overline{M0}$
Set if result is $00; cleared otherwise.

V 0
Cleared

C 0
Cleared

Source Forms: TSTA; TSTB; TST (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TSTA (INH)			TSTB (INH)			TST (EXT)			TST (IND,X)			TST (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	4D	1	OP	5D	1	OP	7D	1	OP	6D	1	OP	18	1
2	OP + 1	—	1	OP + 1	—	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	6D	1
3							OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4							hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5							FFFF	—	1	FFFF	—	1	Y + ff	(Y + ff)	1
6							FFFF	—	1	FFFF	—	1	FFFF	—	1
7													FFFF	—	1

TSX — Transfer from SP to Index Register X — TSX

Operation: IX ⇐ (SP) + $0001

Description: Loads the index register X with one plus the contents of the stack pointer. The contents of the stack pointer remain unchanged. After a TSX instruction, the index register X points at the last value that was stored on the stack.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: TSX

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TSX (INH)		
	Addr	Data	R/W
1	OP	30	1
2	OP + 1	—	1
3	SP	—	1

TXS — Transfer from Index Register X to SP — TXS

Operation: SP ⇐ (IX) – $0001

Description: Loads the stack pointer with the contents of index register X minus one. The contents of index register X remain unchanged.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: TXS

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	TXS (INH)		
	Addr	Data	R/W
1	OP	35	1
2	OP + 1	—	1
3	FFFF	—	1

WAI — Wait for Interrupt — WAI

Operation:

PC ⇐ (PC) + $0001
⇓ (PCL), SP ⇐ (SP) – $0001
⇓ (PCH), SP ⇐ (SP) – $0001
⇓ (IYL), SP ⇐ (SP) – $0001
⇓ (IYH), SP ⇐ (SP) – $0001
⇓ (IXL), SP ⇐ (SP) – $0001
⇓ (IXH), SP ⇐ (SP) – $0001
⇓ (ACCA), SP ⇐ (SP) – $0001
⇓ (ACCB), SP ⇐ (SP) – $0001
⇓ (CCR), SP ⇐ (SP) – $0001

Description:

The program counter is incremented by one. The program counter, index registers Y and X, and accumulators A and B are pushed onto the stack. The CCR is then pushed onto the stack. The stack pointer is decremented by one after each byte of data is stored on the stack.

The MPU then enters a wait state for an integer number of MCU E-clock cycles. While in the wait state, the address/data bus repeatedly runs read bus cycles to the address where the CCR contents were stacked. The MCU leaves the wait state when it senses any interrupt that has not been masked.

Upon leaving the wait state, the MCU sets the I bit in the CCR, fetches the vector (address) corresponding to the interrupt sensed, and instruction execution is resumed at this location.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	1	—	—	—	—

Although the WAI instruction itself does not alter the condition code bits, the interrupt which causes the MCU to resume processing causes the I bit (and the X bit if the interrupt was $\overline{\text{XIRQ}}$) to be set as the interrupt vector is being fetched.

Wait for Interrupt (Continued)

Source Form: WAI

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	WAI (INH)		
	Addr	Data	R/W
1	OP	3E	1
2	OP + 1	—	1
3	SP	Rtn lo	0
4	SP – 1	Rtn hi	0
5	SP – 2	(IYL)	0
6	SP – 3	(IYH)	0
7	SP – 4	(IXL)	0
8	SP – 5	(IXH)	0
9	SP – 6	(A)	0
10	SP – 7	(B)	0
11	SP – 8	(CCR)	0
12 to 12 + n	SP – 8	(CCR)	1
13 + n	Vec hi	Svc hi	1
14 + n	Vec lo	Svc lo	1