

H8/300 CPU
Programming Manual
Ver.-Rev.: 01-03

Contents

Section 4. CPU

- 4.1 Overview
- 4.2 Programming Model
- 4.3 Addressing Mode
- 4.4 Data Types
- 4.5 Instruction Set
- 4.6 Memory Model
- 4.7 Processing States
- 4.8 CPU Basic Timing

Appendix A. Instruction Set Details

Appendix B. Opcode Map

Appendix C. Instruction Set Summary

Section 4. CPU

4.1 Overview

The H8/300 has an 8-bit high-speed CPU employing Hitachi's original architecture with RISC type instruction set. The H8/300 CPU has the following features:

- General purpose registers
 - Sixteen 8-bit general purpose registers or eight 16-bit general purpose registers

- 53 basic instructions
 - Multiplication and division instructions
 - Powerful bit manipulation instructions

- Eight addressing modes
 - Register direct
 - Register indirect
 - Register indirect with displacement
 - Register indirect with post-increment/ pre-decrement
 - Absolute address
 - Immediate
 - Program counter relative
 - Memory indirect

- High-speed operation
 - Most instructions executable in 2 or 4 states
 - 16-bit data addition (in general purpose registers) executed in 200 nsec when $\phi = 10$ MHz

- Low power consumption mode instruction
 - SLEEP

4.2.1 General Purpose Registers

General purpose registers can be used as either data registers or address registers. When they are used as address registers, they function as 16 bit-registers, while they are used as data registers, they function as either 8 bit-registers or 16 bit-registers depending on op-codes. In addition, general purpose register R7 implicitly functions as the stack pointer for interrupt processing and subroutine calls.

4.2.2 Control Registers

Program Counter (PC): 16-bit PC has the address which points to the next instruction to be executed by the CPU.

Condition Code Register (CCR): 8-bit CCR shows the CPU internal status as described below. CCR, including reserved bits 6 and 4, is accessed by CCR manipulation instructions or transfer instructions. CCR can be used as conditions for branch instruction Bcc. Moreover, CCR is not affected by some instructions. (See Appendix A. Instruction Set Details)

Interrupt mask bit (I): When the I bit is set, interrupts other than NMI are masked.

Half carry flag (H): The H flag is set if a carry or borrow is generated by executing an instruction such as ADD.B, ADDX.B, SUB.B, SUBX.B, or CMP.B, and cleared otherwise. The H flag is implicitly used for the DAA and DAS instructions. For the ADD.W, SUB.W, and CMP.W instruction, the H flag is set if a carry or borrow occurs from bit 11, and cleared otherwise.

Negative flag (N): The N flag reflects the MSB of result data, which is handled as a sign bit.

Zero flag (Z): The Z flag is set if result data is zero, and cleared otherwise.

Overflow flag (V): The V flag is set if the result overflows, and cleared otherwise.

Carry flag (C): The C flag is set if a carry is generated from MSB by executing an instruction, and cleared otherwise. The C flag also functions as a bit accumulator for bit manipulation instructions. A carry includes the following events:

- A carry generated by addition, shifting, or rotation
- A borrow generated by subtraction

4.2.3 Initial Values of CPU Internal Registers

PC is initialized by reset exception processing since the PC initial value is loaded from the reset vector. However, general purpose registers and CCR*¹ are not initialized by reset exception processing. Register R7 (SP) is also undefined after reset exception processing. Accordingly, general purpose registers including register R7 (SP) must be initialized immediately after reset exception processing.

Note: 1. An exception is that the I bit of CCR is initialized to 1 after reset exception processing.

4.3 Addressing Modes

The H8/300 CPU supports eight addressing modes as described below.

4.3.1 Register Direct: Rn

An operand is contained in an 8-bit or 16-bit register specified by the register field in the instruction code. The MOV.W, ADD.W, SUB.W, ADDS.W, SUBS.W, MULXU, and DIVXU instructions use word operands.

4.3.2 Register Indirect: @Rn

An operand address is contained in a 16-bit register specified by the register field in the instruction code.

4.3.3 Register Indirect with Displacement: @(disp:16, Rn)

An operand address is calculated by adding a 16-bit displacement contained in the second word (third and fourth bytes) of the instruction code to the contents of a 16-bit register specified by the register field in the instruction code. Register indirect with displacement addressing mode is used only for the MOV instruction. For MOV.W, the effective address should be even.

4.3.4 Register Indirect with Post-increment/ Pre-decrement: @Rn+ / @-Rn

Register indirect with post-increment (@Rn+): This addressing mode is used for MOV instruction. An operand address is contained in a 16-bit register specified by the register field in the instruction code. After instruction execution, 1 or 2 is automatically added to the 16-bit register contents: 1 is added for a byte operand, 2 is added for a word (2 bytes) operand. Note that for word operands, the 16-bit register contents must always be even.

Register indirect with pre-decrement (@-Rn): This addressing mode is used for the MOV instruction. An operand address is contained in a 16-bit register specified by the register field in the instruction code. After instruction execution, 1 or 2 is automatically subtracted from the 16-bit register contents: 1 is decremented for a byte operand, 2 is decremented for a word (2 bytes) operand. Note that for word operands, the 16-bit register contents must always be even.

4.3.5 Absolute Address: @aaaa:8/@aaaa:16

An operand address is specified by 8-bit (@aaaa:8) or 16-bit (@aaaa:16) absolute data in the instruction code. 8-bit absolute addressing mode (@aaaa:8) is used for MOV:B, while 16-bit absolute addressing mode (@aaaa:16) is used for MOV.B, MOV.W, JMP, and JSR. In 8-bit absolute addressing mode, the upper 8 bits of the address are all "1"s (FFH); accessible addresses are 65280 through 65535 (FF00H through FFFFH).

4.3.6 Immediate: #XXX:8/#XXX:16

8-bit immediate data in the 2nd byte of the instruction code is used as an operand. For the ADDS.W and SUBS.W instructions, immediate data (1 or 2) is implicitly contained in their instruction codes. In addition, for bit manipulation instructions, 3-bit immediate data is contained in the second or forth byte of their instruction codes.

4.3.7 Program Counter Relative: @(disp:8, PC)

Program counter relative addressing mode is used for the BRA, BRN, Bcc, and BSR instructions. A branch destination address is calculated by adding an 8-bit displacement in the second byte of the instruction code to the PC, which indicates the start address of the next instruction. The 8-bit displacement is sign extended to 16 bits before addition. The allowable offset from the end of the instruction of the branch destination is in the range of -126 to +128 bytes (-63 to +64 words). Note that the displacement and PC must be even.

4.3.8 Memory Indirect: @@aaaa:8

Memory indirect addressing mode is used for the JMP and JSR instructions. An operand address is specified by 8-bit absolute data in the second byte of the instruction code. The operand contains a jump destination address. The upper 8 bits of the operand address are all "0"s (00H) so that the accessible address range is 0000H through 00FFH. Note that addresses 0000H through 003DH are also used as vector address space.

Table 4-1 shows how the effective address for each addressing mode is calculated. For arithmetic and logical operation instructions, register direct and immediate addressing modes (only for ADD.B, ADDX.B, SUBX.B, CMP.B, AND.B, OR.B, and XOR.B) can be used. Data transfer instructions can use all addressing modes except for PC relative and memory indirect addressing modes. For bit manipulation instructions, register direct, register indirect, and 8-bit absolute addressing mode can be used. Moreover, for specifying bit number (bit position) in the bit manipulation instructions, register indirect (only for BSET, BCLR, BNOT, and BTST) and 3-bit immediate addressing modes are used.

Table 4-1. EA Calculation

No.	Addressing Mode and Instruction Format	EA Calculation	EA
1	Register direct Rn <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 158 74 30 </div> <div style="display: flex; justify-content: space-between; width: 100%;"> OPreg.2reg.1 </div> </div> </div>	None	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 30 </div> <div style="display: flex; justify-content: space-between; width: 100%;"> reg.1reg.2 </div> </div> </div> <p style="text-align: center;">reg.1 and reg. 2 contain operands.</p>
2	Register indirect @Rn <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 157 64 30 </div> <div style="display: flex; justify-content: space-between; width: 100%;"> OPreg. </div> </div> </div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> <div style="text-align: center;">reg. contents (16 bits)</div> </div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> </div>
3	Register indirect with displacement @(disp:16, Rn) <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 157 640 </div> <div style="display: flex; justify-content: space-between; width: 100%;"> OPreg. </div> <div style="text-align: center; margin-top: 5px;">disp</div> </div> </div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> <div style="text-align: center;">reg. contents (16 bits)</div> </div> <div style="border: 1px solid black; padding: 2px; width: 100%; margin-top: 5px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> <div style="text-align: center;">disp</div> </div> <div style="text-align: center; margin-top: 5px;">+</div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> </div>
4	Register indirect with post-increment/pre-decrement Register indirect with post-increment @Rn+ <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 157 64 30 </div> <div style="display: flex; justify-content: space-between; width: 100%;"> OPreg. </div> </div> </div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> <div style="text-align: center;">reg. contents (16 bits)</div> </div> <div style="border: 1px solid black; padding: 2px; width: 100%; margin-top: 5px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 20 </div> <div style="text-align: center;">reg.</div> </div> <div style="text-align: center; margin-top: 5px;">+</div> <div style="text-align: center; margin-top: 5px;">1 or 2</div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> </div>
	Register indirect with pre-decrement @-Rn <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 157 64 30 </div> <div style="display: flex; justify-content: space-between; width: 100%;"> OPreg. </div> </div> </div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> <div style="text-align: center;">reg. contents (16 bits)</div> </div> <div style="border: 1px solid black; padding: 2px; width: 100%; margin-top: 5px;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 20 </div> <div style="text-align: center;">reg.</div> </div> <div style="text-align: center; margin-top: 5px;">-</div> <div style="text-align: center; margin-top: 5px;">1 or 2</div>	<div style="border: 1px solid black; padding: 2px; width: 100%;"> <div style="display: flex; justify-content: space-between; width: 100%;"> 150 </div> </div>

Table 4-1. EA Calculation (cont)

No.	Addressing Mode and Instruction Format	EA Calculation	EA
5	Absolute address @aaaa:8	None	
	@aaaa:16		
6	Immediate #XXX:8	None	Operand is 1-byte immediate data.
	Immediate #XXX:16		Operand is 2-byte immediate data.
7	Program counter relative @(disp:8,PC)		
8	Memory indirect @@aaa:8		

4.4 Data Types

The H8/300 CPU supports three data types: 1-bit data, byte data (8 bits), and word data (16 bits). 1-bit data is handled by bit manipulation instructions and is specified by a bit number out of an 8-bit operand. Byte data can be handled by all instructions other than the ADDS and SUBS instructions. Word data is used by the MOV.W, ADD.W, SUB.W, ADDS.W, SUBS.W, MULXU, and DIVXU instructions. In decimal adjust instructions (DAA and DAS), byte data is handled as 2-digit 4-bit BCD data (packed BCD data).

4.4.1 Data Organization in General Purpose Registers

Figure 4-2 shows data organization in general purpose registers for the various data types.

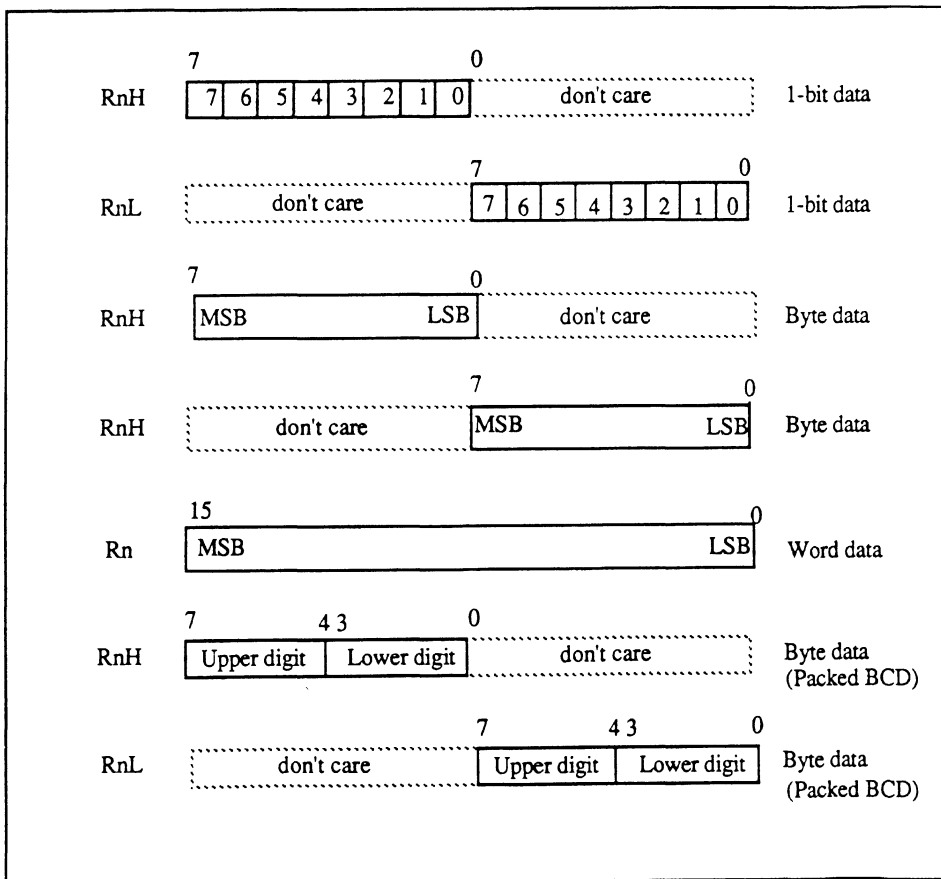


Figure 4-2. Data Organization in General Purpose Registers

4.4.2 Data Organization in Memory

Figure 4-3 shows data organization in memory. The H8/300 CPU can access word (16 bits) data whose upper 8 bits are located at an even address n . However, note that if the upper 8 bits of word data is located at an odd address n , the H8/300 CPU regards the LSB of the address as "0" and accesses word data from address $n-1$. At this time, an address error does not occur. In addition, the stack must always be accessed on a word basis. When CCR is stacked, the same data of CCR is stacked to the upper byte and the lower.

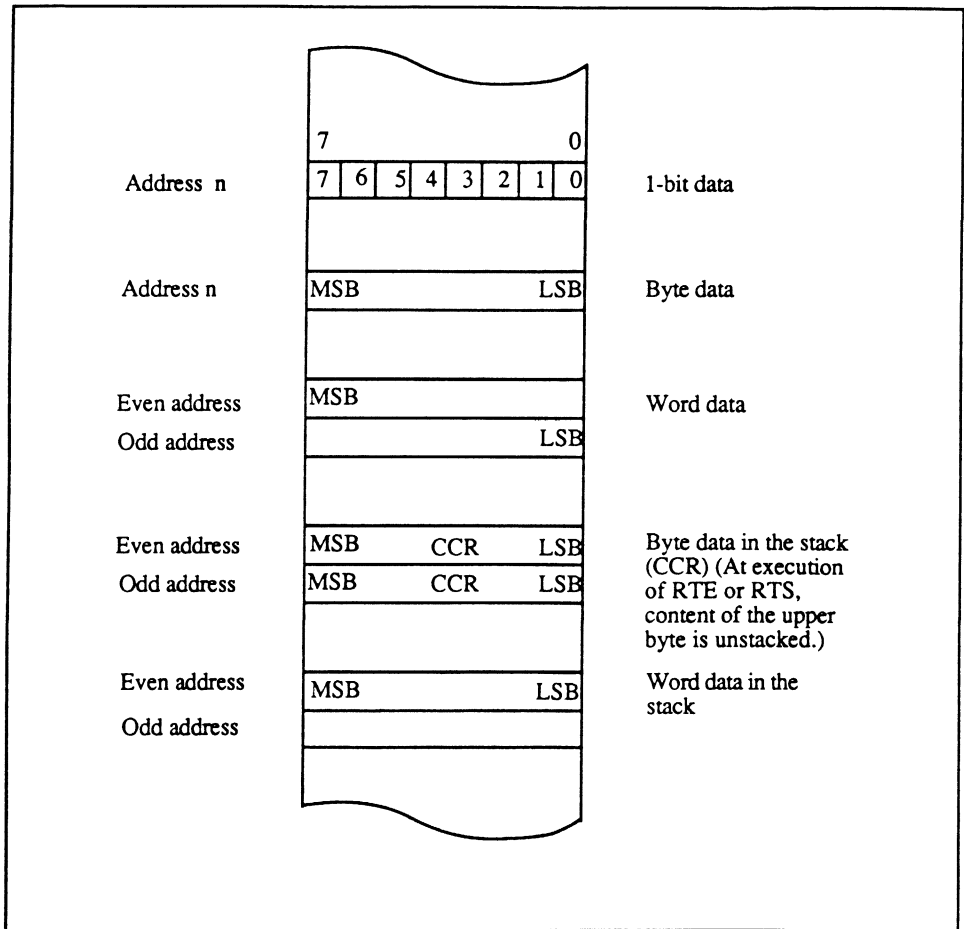


Figure 4-3. Data Organization in Memory

4.5 Instruction Set Summary

The H8/300 CPU has 7 types of instructions as listed below.

Table 4-2. Instruction Types

No.	Instruction Type	No. of Instructions
1	Data transfer	1
2	Arithmetic operation	13
3	Logical operation	3
4	Shift operation	8
5	Bit manipulation	14
6	Branch operation	5
7	System control	8
8	EEPROM write operation	1
	Total	53

4.5.1 Data Transfer Instructions

Table 4-3 shows data transfer instructions and functions.

Table 4-3. Data Transfer Instructions

Instruction	Size	Function
MOV	B, W	Rn -> (EA), (EA) -> Rn, #XXX -> Rn Moves data between general purpose registers, or between a general purpose register and memory, or moves immediate data to a general purpose register. For word data, Rm, @Rm, @(disp:16, Rm), @aaaa:16, @-Rm, and @Rm+ addressing modes can be used. @aaaa:8 can be used only for byte data. However, note that for @-R7 and @R7+ addressing modes, word data must always be specified.
MOVFP	B	@aaaa:16 -> Rn Moves data from memory to a general purpose register synchronous with the E clock.
MOVTP	B	Rn -> @aaaa:16 Moves data from a general purpose register to memory synchronous with the E clock.

Figure 4-4 shows data transfer instruction format.

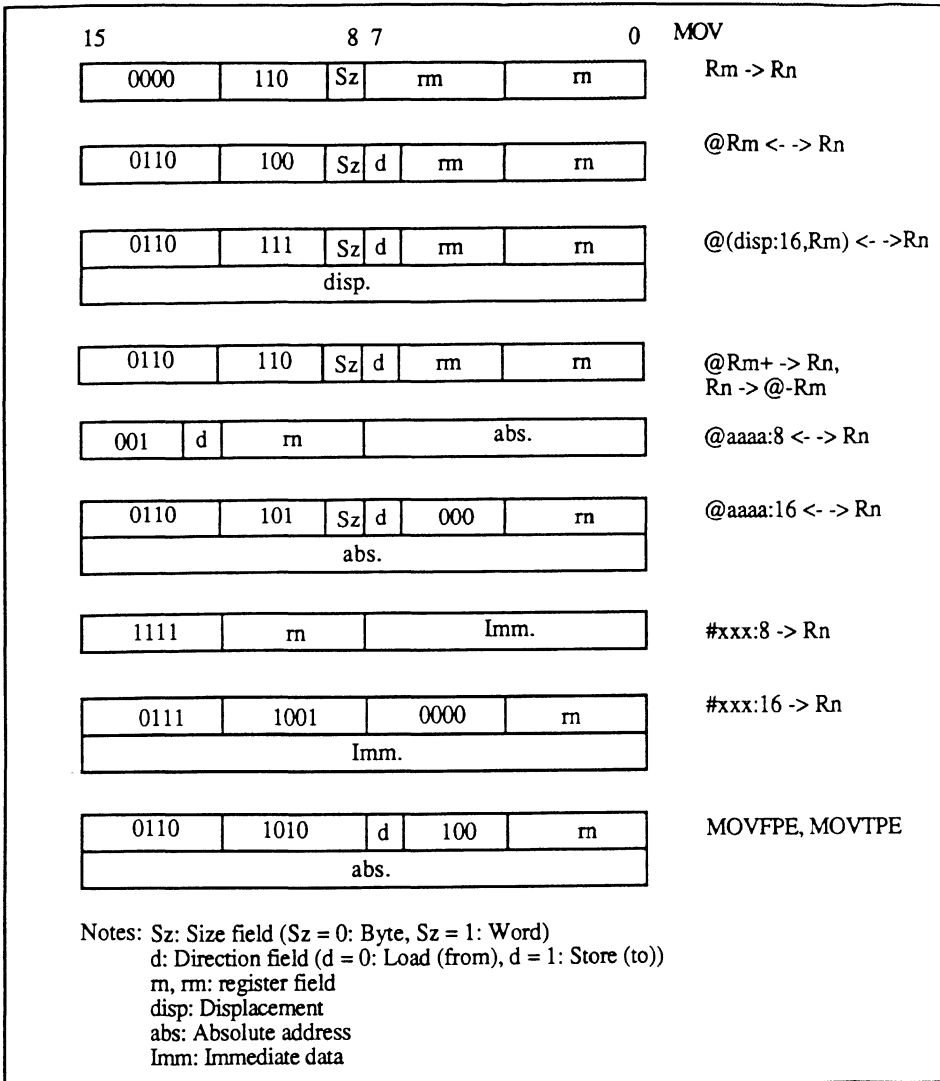


Figure 4-4. Data Transfer Instruction Format

4.5.2 Arithmetic Operation Instructions

Table 4-4 shows arithmetic operation instructions and functions.

Table 4-4. Arithmetic Operation Instructions

Instruction	Size	Function
ADD SUB	B, W	$R_n \pm R_m \rightarrow R_n$, $R_n + \#XXX \rightarrow R_n$ Performs addition or subtraction between general purpose registers, or addition between a general purpose register and immediate data. Note that subtraction between immediate data and a general purpose register cannot be performed. Word data can be handled in addition or subtraction between general purpose registers.
ADDX SUBX	B	$R_n \pm R_m \pm C \rightarrow R_n$, $R_n \pm \#XXX + C \rightarrow R_n$ Performs addition or subtraction with carry between general purpose registers or between a general purpose register and immediate data.
INC DEC	B	$R_n \pm \#1 \rightarrow R_n$ Increments or decrements a general purpose register by 1.
ADDS SUBS	W	$R_n \pm \#XXX \rightarrow R_n$ Performs addition or subtraction between immediate data ("1" or "2") and a general purpose register.
DAA DAS	B	R_n (Decimal Adjustment) $\rightarrow R_n$ Performs decimal adjustment on general purpose register data according to the CCR after arithmetic operation and produces packed BCD data.
MULXU	B	$R_n \times R_m \rightarrow R_n$ Multiplies 8-bit general purpose register data with 8-bit general purpose register data. Multiplication is performed using unsigned arithmetic.

Table 4-4. Arithmetic Operation Instructions (cont)

Instruction	Size	Function
DIVXU	B	$Rn + Rm \rightarrow Rn$ Divides 16-bit general purpose register data by 8-bit general purpose register data.
CMP	B, W	$Rn - Rm, Rn - \#XXX$ Compares data in general purpose registers, or compares data in a general purpose register with immediate data. Word data is used only at comparison of the general purpose register with general purpose register.
NEG	B	$0 - Rn \rightarrow Rn$ Takes two's complement of general purpose register data.

4.5.3 Logical Operation Instructions

Table 4-5 shows logical operation instructions and functions.

Table 4-5. Logical Operation Instructions

Instruction	Size	Function
AND	B	$R_n \wedge R_m \rightarrow R_n$, $R_n \wedge \#XXX \rightarrow R_n$ Performs logical AND operation between general purpose registers or between general purpose register and immediate data.
OR	B	$R_n \vee R_m \rightarrow R_n$, $R_n \vee \#XXX \rightarrow R_n$ Performs logical OR operation between general purpose registers or between a general purpose register and immediate data.
XOR	B	$R_n \oplus R_m \rightarrow R_n$, $R_n \oplus \#XXX \rightarrow R_n$ Performs exclusive OR operation between general purpose registers or between a general purpose register and immediate data.
NOT	B	$\sim R_n \rightarrow R_n$ Takes one's complement of a general purpose register.

4.5.4 Shift Instructions

Table 4-6 shows shift instructions and functions.

Table 4-6. Shift Instructions

Instruction	Size	Function
SHAL, SHAR	B	Rn (shift operation) -> Rn Arithmetically shifts general purpose register data
SHLL SHLR	B	Rn (shift operation) -> Rn Logically shifts general purpose register data
ROTL ROTR	B	Rn rotate operation -> Rn Rotates general purpose register data
ROTXL ROTXR	B	Rn rotate operation -> Rn Rotates general purpose register with C flag

Figure 4-5 shows the instruction format of arithmetic and logical operation instructions, and shift instructions.

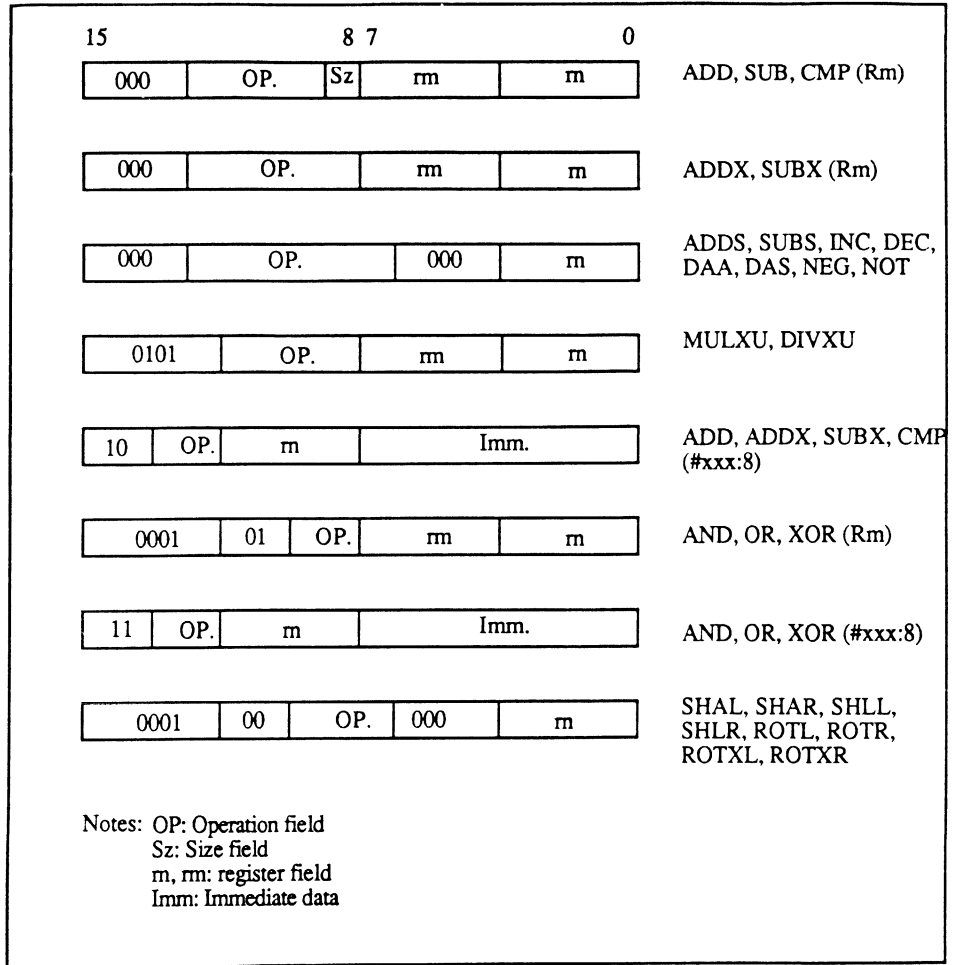


Figure 4-5. Arithmetic/Logical Operation Instruction and Shift Instruction Formats

4.5.5 Bit Manipulation Instructions

Table 4-7 shows bit manipulation instructions and functions.

Table 4-7. Bit Manipulation Instructions

Instruction	Size	Function
BSET	B	1 -> Bit of (EA) Sets a bit within the operand located in a general purpose register or memory. The bit number is specified by immediate data or a general purpose register.
BCLR	B	0 -> Bit of (EA) Clears a bit within the operand located in a general purpose register or memory. The bit number is specified by immediate data or a general purpose register.
BNOT	B	~Bit of (EA) -> Bit of (EA) Inverts a bit of an operand in a general purpose register or memory. The bit number is specified by immediate data or a general purpose register.
BTST	B	~Bit of (EA) -> Z Tests a bit within the operand located in a general purpose register or memory and set the Z flag according to the result. The bit number is specified by immediate data or a general purpose register.
BAND BIAND	B	$C \wedge$ Bit of (EA) -> C, $C \wedge \sim$ Bit of (EA) -> C, Performs logical AND operation between the C bit and a bit within the operand located in a general purpose register or memory (EA), or first inverts a bit of the operand before performing logical AND operation between the C bit and the inverted bit. The bit number is specified by immediate data.

Table 4-7. Bit Manipulation Instructions (cont)

Instruction	Size	Function
BOR BIOR	B	$C \vee \text{Bit of (EA)} \rightarrow C$, $C \vee \sim\text{Bit of (EA)} \rightarrow C$, Performs logical OR operation between the C bit and a bit of the operand located in a general purpose register or memory (EA), or first inverts a bit of the operand and then perform logical OR operation between the C bit and the inverted bit. The bit number is specified by immediate data.
BXOR BIXOR	B	$C \oplus \text{Bit of (EA)} \rightarrow C$, $C \oplus \sim\text{Bit of (EA)} \rightarrow C$, Performs exclusive OR operation between the C bit and a bit of the operand located in a general purpose register or memory (EA), or first inverts a bit of the operand and then perform exclusive OR operation between the C bit and the inverted bit. The bit number is specified by immediate data.
BLD BILD	B	Bit of (EA) $\rightarrow C$, \sim Bit of (EA) $\rightarrow C$ Loads a bit of the operand located in a general purpose register or memory (EA) into the C flag, or first inverts a bit of the operand and then loads the inverted bit into the C flag. The bit number is specified by immediate data.
BST BIST	B	$C \rightarrow \text{Bit of (EA)}$, $\sim C \rightarrow \text{Bit of (EA)}$ Stores the C flag into a bit of the operand located in a general register or memory (EA), or first inverts the C flag and then stores the inverted C flag into a bit of the operand. The bit number is specified by immediate data.

Figure 4-6 shows bit manipulation instruction formats.

15	8	7				0	
0110	0	OP.	0	Imm.	m		BSET, BCLR, BNOT, BTST Operand: Register direct (Rn) Bit No.: Immediate (#xxx:3)
0111	0	OP.	m		m		Operand: Register direct (Rn) Bit No.: Register direct (Rm)
0110	10	OP.	m	0000			Operand: Register indirect (@Rn) Bit No.: Immediate (#xxx:3)
0110	0	OP.	0	Imm.	0000		
0110	10	OP.	m	0000			Operand: Register indirect (@Rn) Bit No.: Register direct (Rm)
0111	0	OP.	m	0000			
0110	10	OP.	abs.				Operand: Absolute address (@aaaa:8) Bit No.: Immediate (#xxx:3)
0110	0	OP.	0	Imm.	0000		
0110	10	OP.	abs.				Operand: Absolute address (@aaaa:8) Bit No.: Register direct (Rm)
0111	0	OP.	m	0000			

Notes: OP: Operation field
d: Direction field (d = 0: Load (from), d = 1: Store (to))
m, mm: register field
abs: Absolute address
Imm: Immediate data

Figure 4-6. Bit Manipulation Instruction Formats

15							8 7			0	BAND, BOR, BXOR, BLD, BST
011	d	0	OP.	0	Imm.	rn					Operand: Register direct (Rn) Bit No.: Immediate (#xxx:3)
0110	10		OP.	rn		0000					Operand: Register indirect (@Rn) Bit No.: Immediate (#xxx:3)
011	d	0	OP.	0	Imm.	0000					
0110	10		OP.	abs.							Operand: Absolute address (@aaaa:8) Bit No.: Immediate (#xxx:3)
011	d	0	OP.	0	Imm.	0000					
											BIAND, BIOR, BIXOR, BILD, BTST
011	d	0	OP.	1	Imm.	rn					Operand: Register direct (Rn) Bit No.: Immediate (#xxx:3)
0110	10		OP.	rn		0000					Operand: Register indirect (@Rn) Bit No.: Immediate (#xxx:3)
011	d	0	OP.	1	Imm.	0000					
0110	10		OP.	abs.							Operand: Absolute address (@aaaa:8) Bit No.: Immediate (#xxx:3)
011	d	0	OP.	1	Imm.	0000					

Notes: OP: Operation field
d: Direction field (d = 0: Load (from), d = 1: Store (to))
rn, rn: register field
abs: Absolute address
Imm: Immediate data

Figure 4-6. Bit Manipulation Instruction Formats (cont)

4.5.6 Branch Instructions

Table 4-8 shows branch instructions and functions.

Table 4-8. Branch Instructions

Instruction	Size	Function			
Bcc	-	Branches to a specified address if the condition specified in cc is true.			
		Mnemonic	cc	Condition	Test
		BRA (BT)	0000	Always (True)	1
		BRN (BF)	0001	Never (False)	0
		BHI	0010	High	$\sim(C + Z)$
		BLS	0011	Low or Same	$C + Z$
		BCC (BHS)	0100	Carry Clear (High or Same)	$\sim C$
		BCS (BLO)	0101	Carry Set (Low)	C
		BNE	0110	Not Equal	$\sim Z$
		BEQ	0111	Equal	Z
		BVC	1000	Overflow Clear	$\sim V$
		BVS	1001	Overflow Set	V
		BPL	1010	Plus	$\sim N$
		BMI	1011	Minus	N
		BGE	1100	Greater Than or Equal	$\sim(N ++ V)$
		BLT	1101	Less Than	$N + V$
		BGT	1110	Greater Than	$\sim(Z + (N ++ V))$
		BLE	1111	Less Than or Equal	$Z + (N ++ V)$
JMP	-	Always jumps to a specified address			
BSR JSR	-	Branches to a subroutine at a specified address.			
RTS	-	Returns from a subroutine.			

Figure 4-7 shows branch instruction formats.

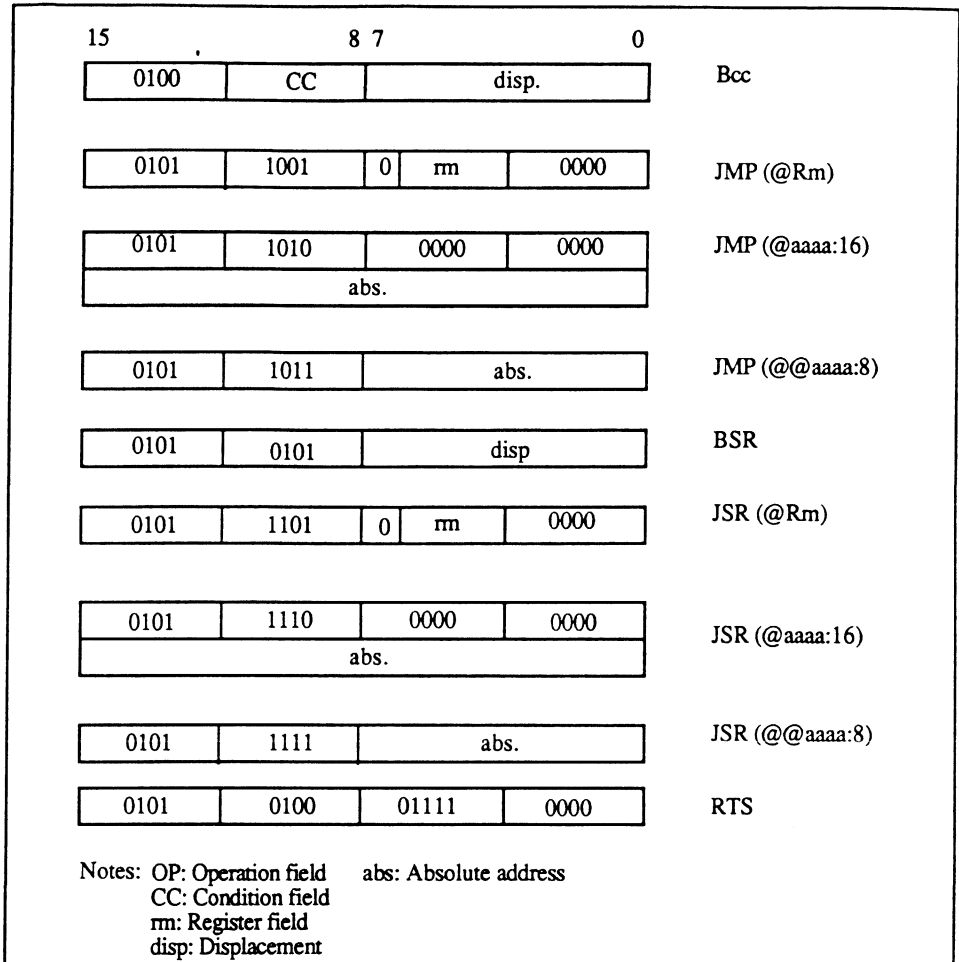


Figure 4-7. Branch Instruction Formats

4.5.7 System Control Instructions

Table 4-9 shows system control instructions and functions.

Table 4-9. System Control Instructions

Instruction	Size	Function
RTE	-	Returns from an interrupt service routine.
SLEEP	-	Enters sleep mode and waits for interrupt generation.
LDC	B	Rn -> CCR, #XXX -> CCR Loads general purpose register data or immediate data into the CCR.
STC	B	CCR -> Rn Stores the CCR in a general purpose register.
ANDC	B	CCR ^ #XXX -> CCR Performs a logical AND operation between the CCR and immediate data.
ORC	B	CCR V #XXX -> CCR Performs a logical OR operation between the CCR and immediate data.
XORC	B	CCR ++ #XXX -> CCR Performs an exclusive OR operation between the CCR and immediate data.
NOP	-	Performs no operation. Only PC is incremented.

Figure 4-8 shows the system control instruction formats.

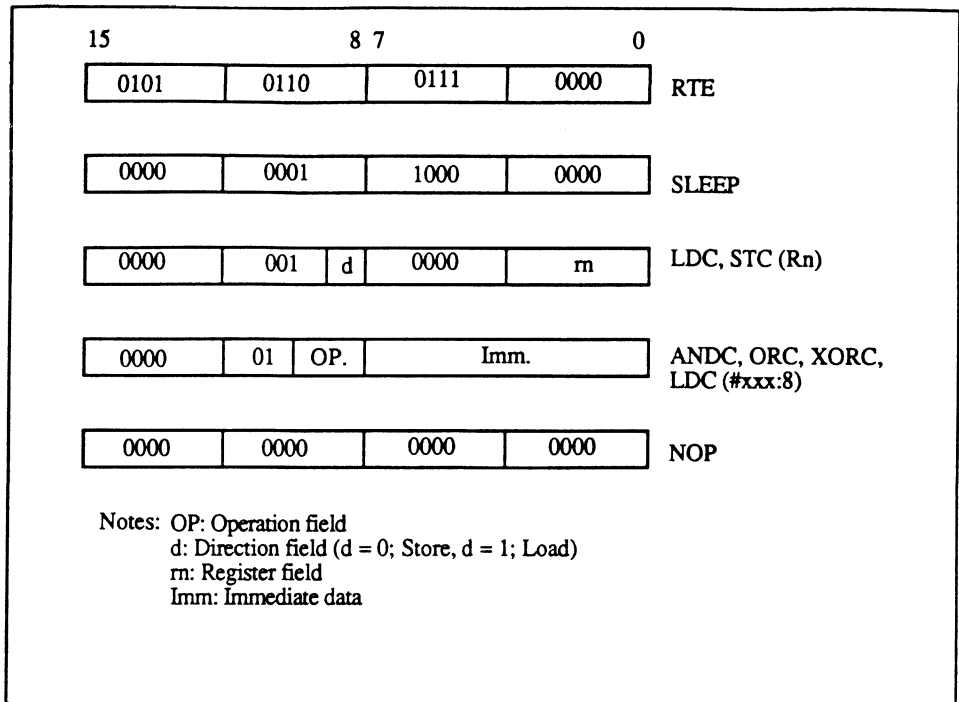


Figure 4-8. System Control Instruction Formats

4.5.8 EEPROM Write Instruction

Table 4-10 shows the EEPROM write instruction and function.

Table 4-10. EEPROM Write Instruction

Instruction	Size	Function
EEPMOV	B	<p>@R5+ -> @R6+</p> <p>EEPMOV is a dedicated EEPROM write instruction. Data in memory, whose start address is specified by R5 and whose byte size is specified by R4L, is loaded to EEPROM whose start address is specified by R6. The next instruction can be executed after the data has been loaded and EEPROM write has completed.</p>

Figure 4-9 shows the EEPROM instruction format.

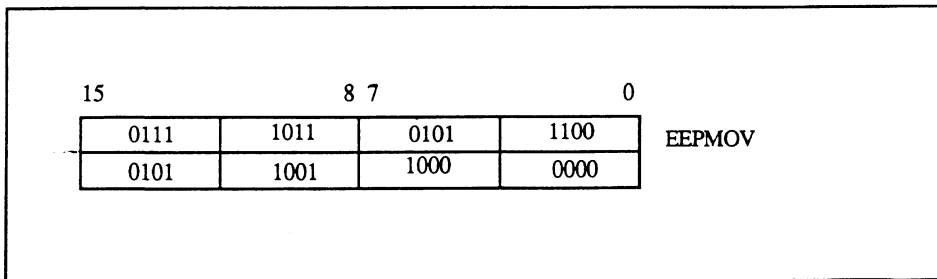


Figure 4-9. EEPMOV Instruction Format

- *1. The EEPMOV instruction acts as a simple block data transfer instruction when MCU has no EEPROM.
- *2. The next instruction can be executed immediately after the transfer completion.

4.6 Memory Model

The H8/300 CPU can support a maximum of 64 K bytes of memory space which can be used for program code and data storage areas.

4.7 Processing States

The H8/300 CPU has three processing states: normal state, exception processing state, and low power consumption state as shown in Figure 4-9. Figure 4-10 shows the transition diagram between each state.

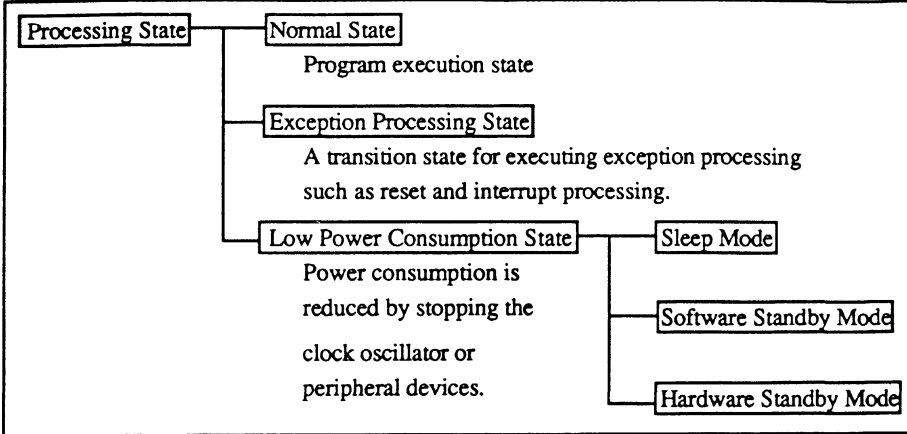


Figure 4-9. Processing States

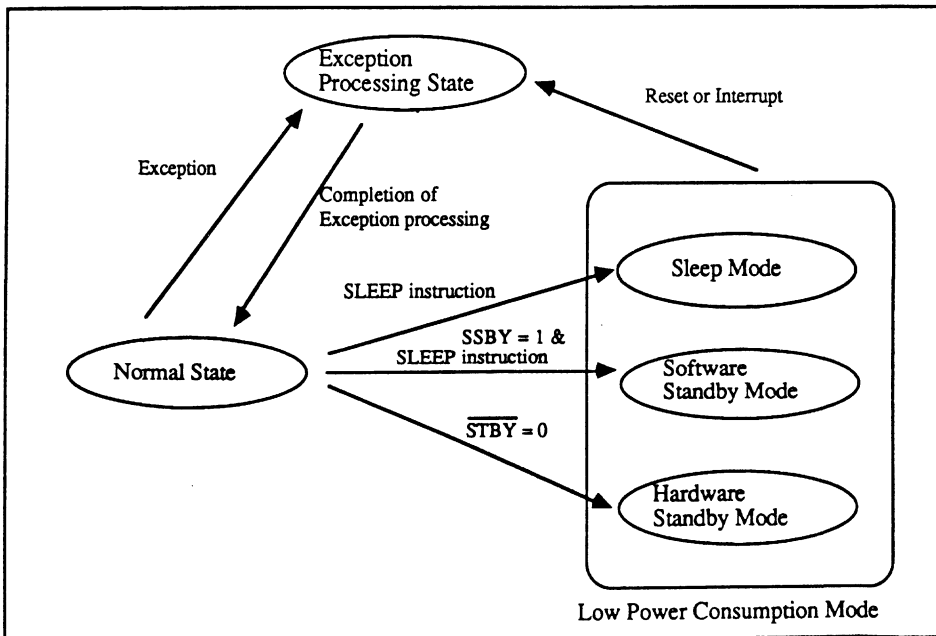


Figure 4-10. Processing State Transition Diagram

4.7.1 Normal State

In normal state, the H8/300 CPU executes a program sequentially. Normal state has two modes: normal and interrupt modes, in which usable resources are almost the same. Branching to or returning from a subroutine is achieved by referring to the stack pointer (SP:R7) and the program counter (PC).

4.7.2 Exception Processing State

Exception processing state is a transition state before reset or interrupt processing begins. All interrupts other than reset push the PC and CCR onto the stack according to the SP.

4.7.3 Low Power Consumption State

Sleep Mode: Sleep mode is entered by executing the SLEEP instruction; the CPU halts operation just after the SLEEP instruction and the CPU internal status is retained. Sleep mode is cancelled by reset or an interrupt. If a reset or interrupt exception occurs, the CPU enters normal state via exception processing state.

Software Standby Mode: Software standby mode is entered by executing the SLEEP instruction after the SSBY bit of SBYCR has been set. In software standby mode, the CPU and all on-chip peripheral devices halt operation. However, the CPU internal status and internal RAM are retained.

Hardware Standby Mode: Hardware standby mode is entered by bringing the STBY pin to low. In this mode, CPU and all on-chip peripheral devices halt operation.

See "6. Low Power Consumption Mode" for details.

4.8 CPU Basic Timing

The H8/300 CPU operates based on the ϕ clock, in which one rising edge to the next rising edge period is called a state. A memory cycle or bus cycle is normally comprised of 2 or 3 states. On-chip memory, internal peripheral device, and external device access timings are shown below.

4.8.1 On-chip Memory (RAM, ROM, and EEPROM) Access Timing

On-chip memory is accessed in 2 states. A 16-bit data bus enables byte or word size access. Figure 4-11 shows on-chip memory access timing.

4.8.2 Internal Peripheral Devices/External Devices Access Timing

Internal peripheral devices and external devices are accessed in 3 states by using an 8-bit data bus. Word data and word instruction code are accessed in two groups of one-byte each. Figures 4-11 and 4-12 show internal memory and peripheral device access timings, respectively. Figures 4-13 and 4-14 show external devices access timings.

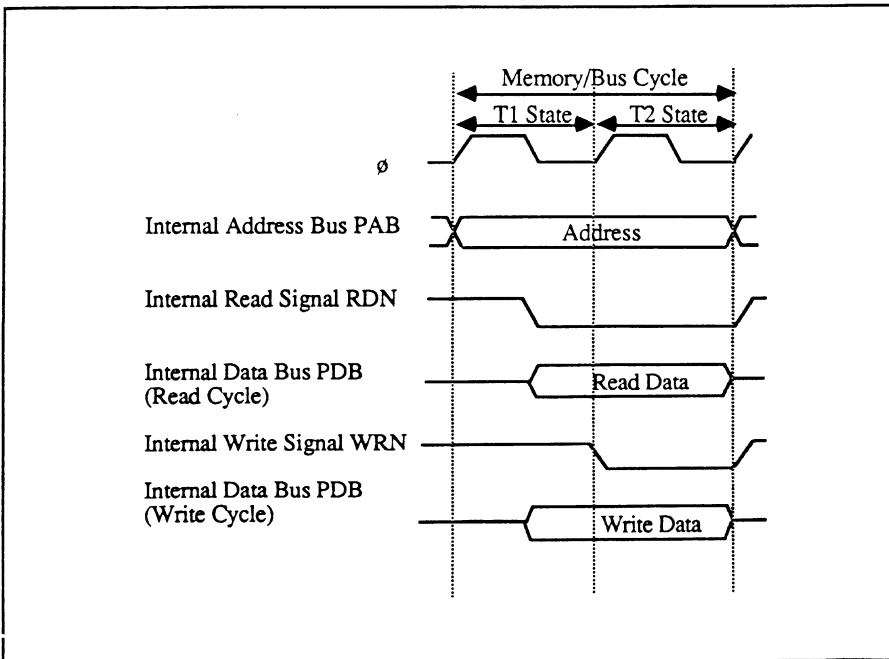


Figure 4-11. Internal Memory Access Timing

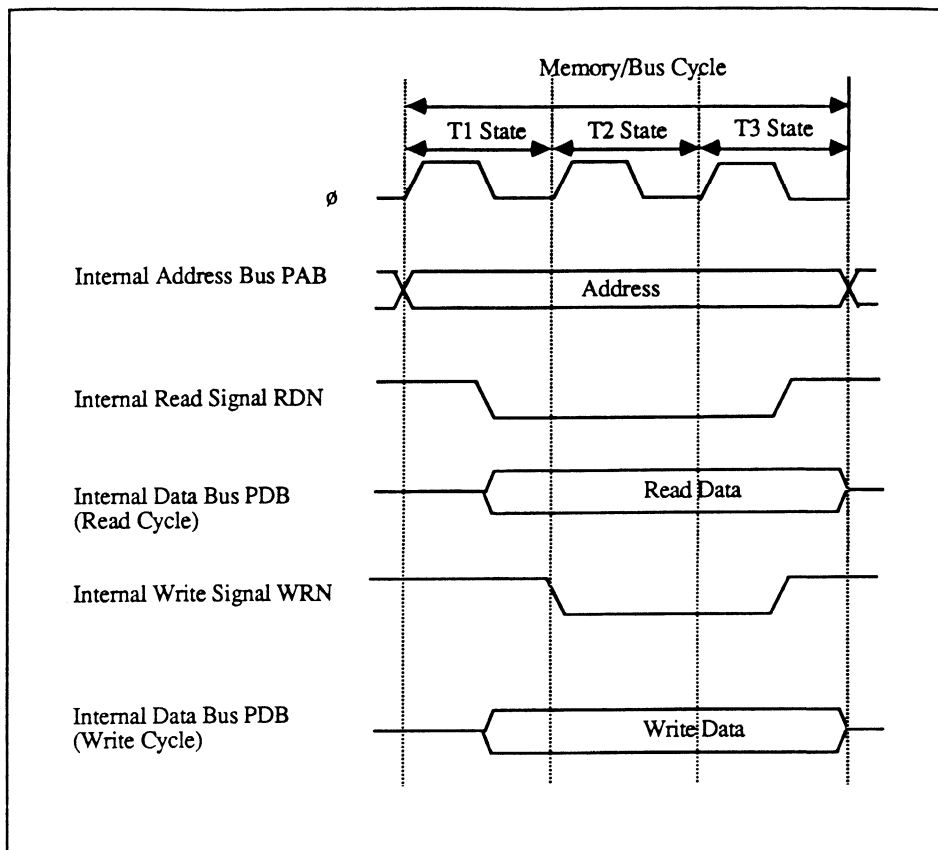


Figure 4-12. Internal Peripheral Device Access Timing

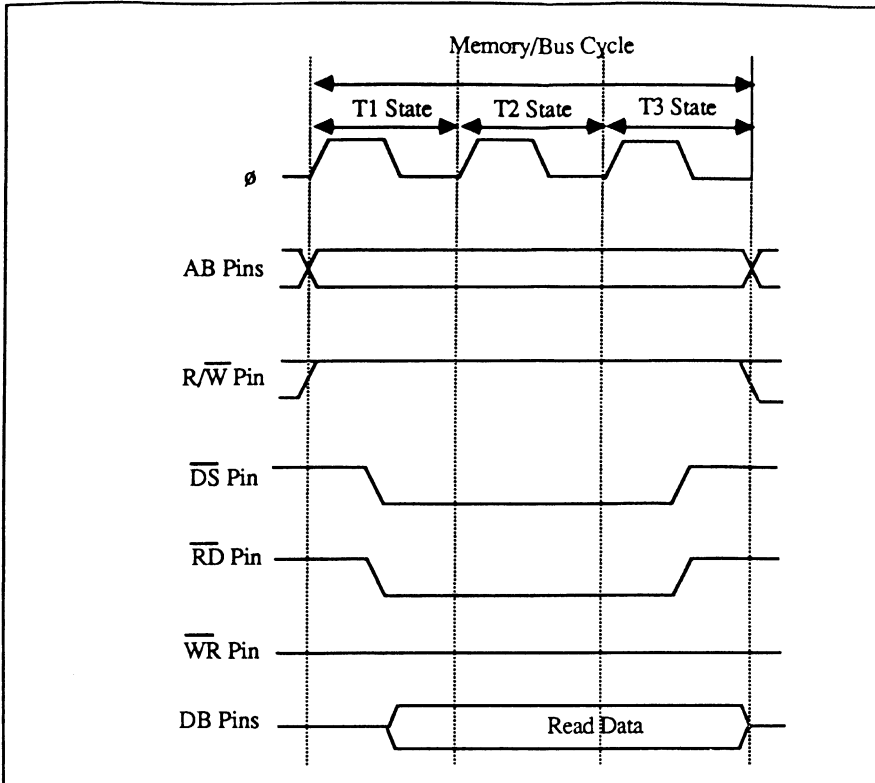


Figure 4-13. External Device Access Timing (Read Cycle)

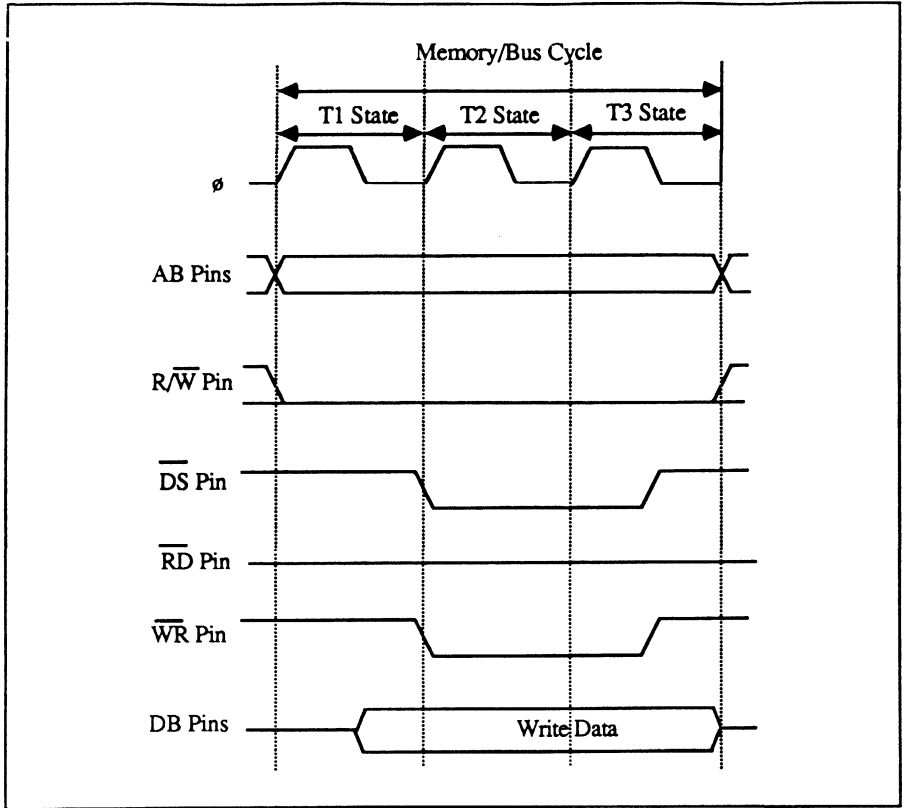


Figure 4-14 External Device Access Timing (Write Cycle)

4.9 6800 Family Bus Interface

The H8/300 CPU provides two transfer instructions synchronous with E clock (MOVFPPE and MOVTPPE) and an E clock pin to facilitate 6800 family bus interface in expansion modes. The E clock is produced by dividing ϕ clock by eight and can be output when the DDR0 pin of port 8 is set to 1 (After reset, E clock is output.).

When the CPU executes the MOVFPPE or MOVTPPE instruction, address bus, IOOE and data bus are output in the same timing as in normal access. However, DS, RD, and WR are asserted after the falling edge of the E clock is detected as shown in Figures 4-15 and 4-16. In addition, MOVFPPE and MOVTPPE execution cycles are 8-15 states and cannot be fixed. Moreover, Tw state by the WAIT pin cannot be inserted during MOVFPPE and MOVTPPE instruction cycles.

When interfacing the H8/300 CPU and a 6800 family peripheral device, the WR pin of the H8/300 CPU must be connected to the R/W pin of the 6800 family peripheral device. All pins other than WR can be connected in the same way as in normal peripheral device interface. Needless to say, the system must be designed so that all peripheral device timing specifications are satisfied.

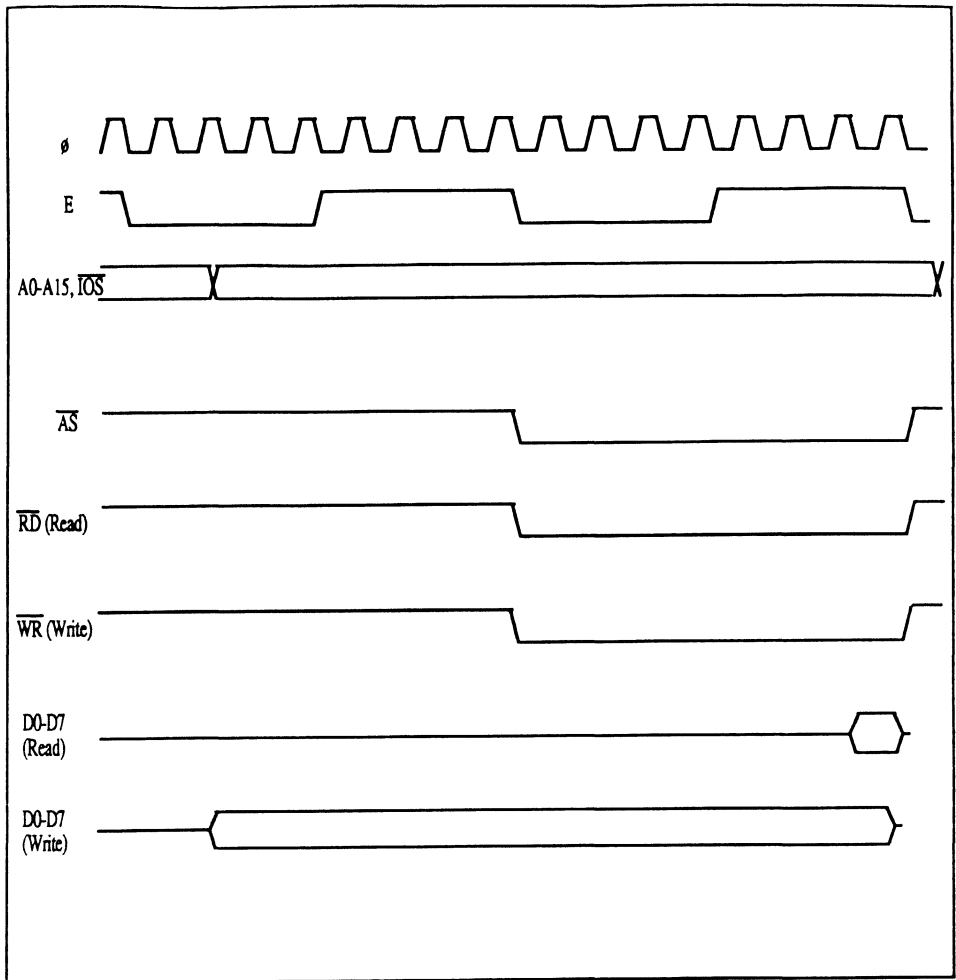


Figure 4-15. FMOVFP or FMOVTP Instruction Cycle Timing in Expansion Mode (Maximum Synchronization)

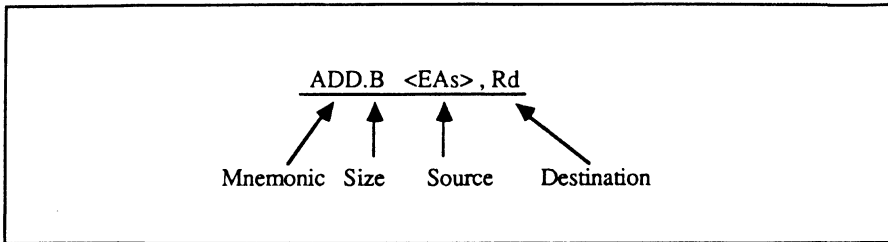
Appendix A. Instruction Set Details

This section describes the H8/300 CPU instruction set.

A-1. Instruction Set Description

Items used in the instruction set details are outlined below.

Assembler Format:



Byte (B) and/or word (W) operand can be selected in the size field according to the instruction. In addition, the size field can be omitted for an instruction whose operand size is fixed to byte (B) or word (W).

<EA> indicates that multiple addressing modes can be specified in the EA field. The H8/300 CPU supports the addressing modes listed below.

Symbol	Addressing Mode
R	Register direct
@R	Register indirect
@(disp:16, R)	Register indirect with displacement
@R+, @-R	Register indirect with post-increment or pre-decrement
@aaaa:8/16	Absolute address (8 or 16 bits)
#XXX:3/8/16	Immediate (3, 8 or 16 bits)
@(disp: 8, PC)	Program counter relative
@@aaaa:8	memory indirect

Operation: Symbols used in the operation field is listed below.

Symbol	Description
PC	Program counter
SP	Stack pointer (R7)
CCR	Condition code register
Z	Zero flag in CCR
C	Carry flag in CCR
Rs, Rd, Rn	General purpose register (8-bit: R0H-R7L or 16 bits: R0-R7)
disp	Displacement
->	Direction of operand transfer or status transition
+	Addition
-	Subtraction
x	Multiplication
/	Division
^	Logical AND
v	Logical OR
++	Exclusive OR
~	One's complement
() or < >	Contents of effective address

Condition Code: Symbols used in condition codes are listed below.

Symbol	Description
--------	-------------

S	Set normally, that is: I: Set if an interrupt is masked, cleared otherwise. H: Set if carry or borrow is generated from bit 3 or bit 11, cleared otherwise. N: Set if result is negative, cleared otherwise. Z: Set if result is zero, cleared otherwise. V: Set if result overflows, cleared otherwise. C: Set if carry or borrow is generated from bit 7 or bit 15, cleared otherwise.
---	--

S*	See explanation
----	-----------------

0	Cleared to 0
---	--------------

1	Set to 1
---	----------

*	Undefined
---	-----------

U	Unaffected
---	------------

Instruction Format: Symbols used in the instruction format is listed below.

Symbol	Description
Imm	Immediate data (3, 8 or 16 bits)
abs	Absolute address (8 or 16 bits)
disp	Displacement (8 or 16 bits)
rs, rd, rn	Register number (3 or 4 bits). rs corresponds to Rs and @Rs addressing modes, rd corresponds to Rd and @Rd addressing modes, and rn corresponds to Rn addressing mode, respectively.

When rs, rd, and rn are used as an address register for @R, @(disp:16, R), @-R, or @R+ addressing mode, they function as 16-bit registers specified by the 3-bit register field as shown below.

On the other hand, when rs, rd, and rn are used as a data register, they function as either a 16-bit register specified by a 3-bit register field for word data or 8-bit register specified by 4-bit register field for byte data. In the 4-bit register field, the lower 3 bits specify the register number and the MSB specifies the upper or lower byte of the register.

16-bit Register Specification:

3-bit r Field	Register
000	R0
001	R1
:	:
:	:
111	R7

8-bit Register Specification:

4-bit r Field	Register
0000	R0H
0001	R1H
:	:
:	:
0111	R7H
1000	R0L
1001	R1L
:	:
:	:
1111	R7L

Bit Data Access in Bit Manipulation Instructions: Bit data can be accessed as the *n*-th bit (*n* = 0 to 7) of a byte operand in register or memory. The bit number *n* is specified by 3-bit immediate data or the lower 3 bits of a general purpose register. Examples, whose addressing mode are register or memory, are shown below.

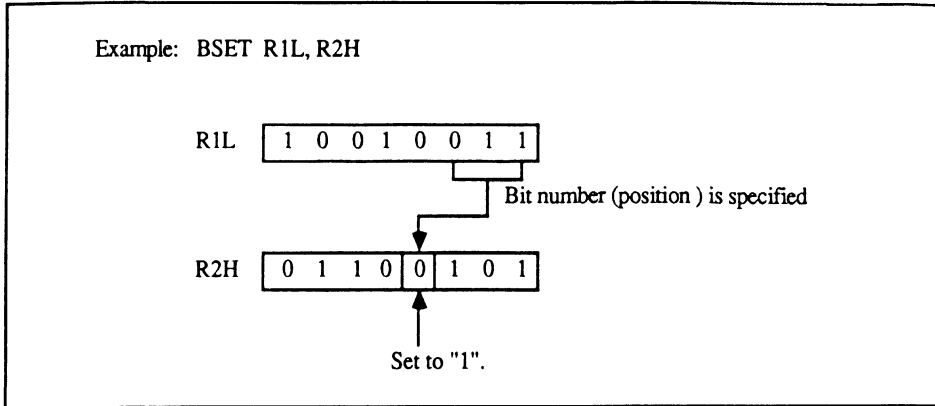


Figure A-1. Bit Data Access in Bit Manipulation Instruction 1

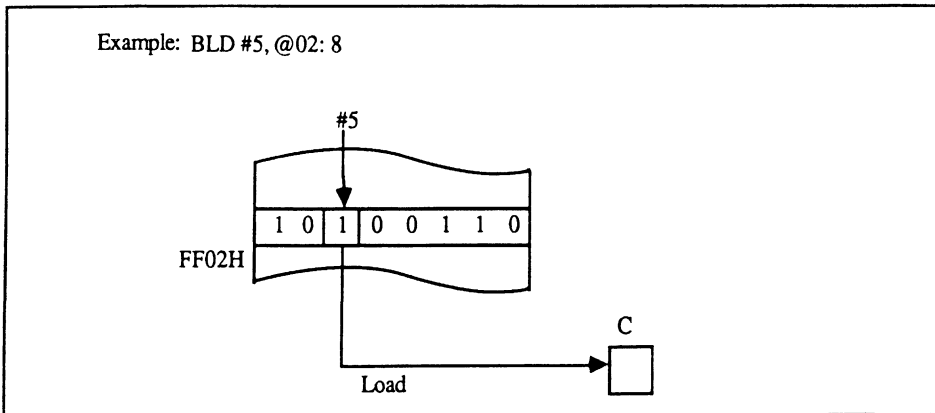


Figure A-2. Bit Data Access in Bit Manipulation Instruction 2

Execution States: Execution states indicates the number of execution states when the instruction and operand are stored in internal memory. When the instruction or operand is stored in area other than internal memory area, the number of execution states increases as shown below.

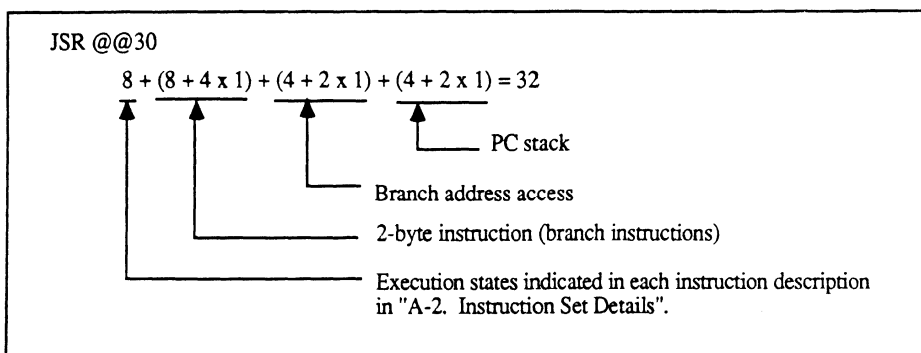
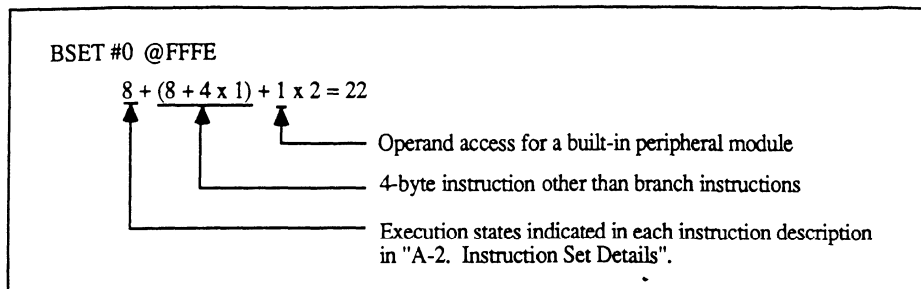
Access Condition		Number of Increased Execution States	
Operand	On-chip peripheral devices	Byte data	1
		Word data	4
	External device	Byte data	1 + m
		Word data	4 + 2m
Instruction	Instruction other than branch instruction	External memory 2-byte instruction	4 + 2m
		4-byte instruction	8 + 4m
	Branch instruction	2-byte instruction	8 + 4m
		4-byte instruction	12 + 6m

Notes: 1. m = number of Tw states that might be inserted in external device access cycle

2. Operand data is accessed twice in BSET, BCLR, BNOT, BST, BIST instructions.
3. In memory indirect addressing mode (@@aaaa:8), (4 + 2m) states should be added for word operand access (branch address access) in mode 1.
4. If the stack is located in an external device, (4 + 2m) states should be added for BSR, JSR, and RTS instruction cycles since BSR, JSR, and RTS performs word operand access for PC pushing or popping.

5. RTE performs word operand twice for CCR and PC popping. Accordingly, if the stack is located in an external device, $(8 + 4m)$ states should be added.

Examples: When the stack is located in an external device and one Tw state is inserted for an external device access cycle in mode 1



6. If R4L is specified as n in the EEPMOV instruction, source and destination operands are accessed $(n + 1)$ times each.

7. *1 = not added for MOVFE and MOVTPE

A-2. Instruction Set Details

A.2.1 ADD (Add Binary)

Assembler Format: ADD.B <EAs>, Rd

Operation: (Destination) + (Source) -> Destination

Description: Adds the source operand to the destination operand, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I - H - N Z V C
U U S U S S S S

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	ADD.B	#:8, Rd	8	rd	Imm		2
Register direct	ADD.B	Rs, Rd	0	8	rs	rd	2

Notes:

A.2.2 ADD (Add Binary)

Assembler Format: ADD.W Rs, Rd

Operation: (Destination) + (Source) -> (Destination)

Description: Adds the source operand to the destination operand, and loads the result to the destination.

Operand Size: Word

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S	S	S

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	ADD.W	Rs, Rd	0 9	0 rs	0 rd	2	

Notes:

A.2.3 ADDX (Add with Extended Carry)

Assembler Format: ADDX.B <EAs>, Rd

Operation: (Destination) + (Source) + C -> (Destination)

Description: Adds the source operand to destination operand with C flag, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S*	S	S

S*: unchanged if result is zero,
cleared to 0 otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	ADDX.B	#:8, Rd	9 rd	Imm			2
Register direct	ADDX.B	Rs, Rd	0 E	rs rd			2

Notes: None

A.2.4 ADDS (Add with Sign Extension)

Assembler Format: ADDS.W #1, Rd
 ADDS.W #2, Rd

Operation: (Destination) + 1 -> (Destination)
 (Destination) +2 -> (Destination)

Description: Adds immediate data "1" or "2" to destination operand, and loads the result to the destination. ADDS execution does not affect condition codes.

Operand Size: Word

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register direct	ADDS.W	#1, Rd	0 B	0	0 rd		2
Register direct	ADDS.W	#2, Rd	0 B	8	0 rd		2

Notes: ADDS cannot handle byte data.

A.2.5 AND (AND Logical)

Assembler Format: AND.B <EAs>, Rd

Operation: (Destination) \wedge (Source) \rightarrow (Destination)

Description: Performs a logical AND operation between the source operand and destination operand, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	U

Operand Format and Number of Execution States:

Instruction Format								
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States	
			1st Byte	2nd Byte	3rd Byte	4th Byte		
Immediate	AND.B	#:8, Rd	E	rd	Imm			2
Register direct	AND.B	Rs, Rd	1	6	rs	rd		2

Notes: None

A.2.6 ANDC (AND Control Register)

Assembler Format: ANDC #:8, CCR

Operation: (CCR) \wedge (Immediate data) \rightarrow (CCR)

Description: Performs a logical AND operation between the immediate data and CCR, and loads the result to the CCR. All interrupts are not detected at completion of this instruction.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
S*	S*	S*	S*	S*	S*	S*	S*

S*: Same as the corresponding bit of the result.

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Instruction				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	ANDC	#:8, CCR	0	6	Imm		2

Notes: None

A.2.7 BAND (Bit AND)

Assembler Format: BAND #:3, <EAd>

Operation: $C \wedge (\text{Bit number of } \langle \text{Destination} \rangle) \rightarrow C$

Description: Performs a logically AND operation between the C flag and a bit of destination operand, and loads the result into the C flag (Figure A-3). The bit number is specified by 3-bit immediate data. The destination operand is not affected by the BAND instruction.

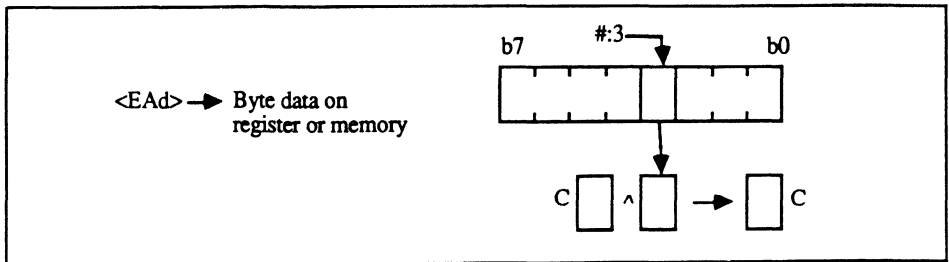


Figure A-3. BAND Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: C set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Addressing Mode	Operand Mnemonic Format	Instruction Format								Exec. States
		1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte	8th Byte	
Register direct	BAND #:3, Rd	7	6	0	Imm rd					2
Register indirect	BAND #:3, @Rd	7	C	0	rd	0	7	6	0 Imm 0	6
Absolute address	BAND #:3, @aaaa:8	7	E	abs		7	6	0 Imm 0		6

Notes: "Addressing mode" above specifies effective address <EAd>.

A.2.8 Bcc (Branch Conditionally)

Assembler Format: Bcc disp.

Operation: If cc is true then PC + disp -> PC
else go to next instruction.

Description: Branches to an address (branch destination) by adding displacement to the PC if the condition specified in the cc field is true; executes the next instruction otherwise. Displacement is 7-bit signed data, while the PC points to the next instruction address. Accordingly, the branch destination is within the range of -126 bytes to +128 bytes from Bcc.

Table A-1 lists condition codes and their mnemonics.

Table A-1. Conditions for Bcc

Mnemonic	Condition	Code	Conditional Expression	Signed Condition Code
T	True (BRA)	0000	1	
F	False (BRN)	0001	0	
HI	High	0010	$\sim C \cdot \sim Z$ [$\sim (C + Z)$]	$x > y$ (Unsigned)
LS	Lower or same	0011	$C + Z$	$x \leq y$ (Unsigned)
CC	Carry Clear (High or Same)	0100	$\sim C$	$x \geq y$ (Unsigned)
CS	Carry Set (Low)	0101	C	$x < y$ (Unsigned)
NE	Not Equal	0110	$\sim Z$	$x \neq y$ (Unsigned/Signed)
EQ	Equal	0111	Z	$x = y$ (Unsigned/Signed)
VC	Overflow Clear	1000	$\sim V$	
VS	Overflow Set	1001	V	
PL	Plus	1010	$\sim N$	
MI	Minus	1011	N	
GE	Greater Than or Equal	1100	$N \cdot V + \sim N \cdot \sim V$ [$\sim (N ++ V)$]	$x \geq y$ (Signed)
LT	Less Than	1101	$N \cdot \sim V + \sim N \cdot V$ [$N ++ V$]	$x < y$ (Signed)
GT	Greater Than	1110	$N \cdot V \cdot \sim Z + \sim N \cdot \sim V \cdot Z$ [$\sim (Z + (N ++ V))$]	$x > y$ (Signed)
LE	Less Than or Equal	1111	$Z + N \cdot \sim V + \sim N \cdot V$ [$Z + (N ++ V)$]	$x \leq y$ (Signed)

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
PC relative	BT (BRA)	disp	4	0	disp		4
PC relative	BF (BRN)	disp	4	1	disp		4
PC relative	BHI	disp	4	2	disp		4
PC relative	BLS	disp	4	3	disp		4
PC relative	Bcc (BHS)	disp	4	4	disp		4
PC relative	BCS (BL0)	disp	4	5	disp		4
PC relative	BNE	disp	4	6	disp		4
PC relative	BEQ	disp	4	7	disp		4
PC relative	BVC	disp	4	8	disp		4
PC relative	BVS	disp	4	9	disp		4
PC relative	BPL	disp	4	A	disp		4
PC relative	BMI	disp	4	B	disp		4
PC relative	BGE	disp	4	C	disp		4
PC relative	BLT	disp	4	D	disp		4
PC relative	BGT	disp	4	E	disp		4
PC relative	BLE	disp	4	F	disp		4

Notes: 1. Branch destination must always be an even address.

2. Machine codes for BT and BF are the same as those for BRA and BRN.

In addition, BF (BRN) functions in the same way as two NOP instructions.

A.2.9 BCLR (Bit Clear)

Assembler Format: BCLR #:3, <EAd>
BCLR Rn, <EAd>

Operation: 0 -> (<Bit number> of <Destination>)

Description: Clears a bit in destination operand (Figure A-4). The bit number is specified by 3-bit immediate data or the lower 3 bits of a general purposer register. Condition codes are not affected by the BCLR instruction.

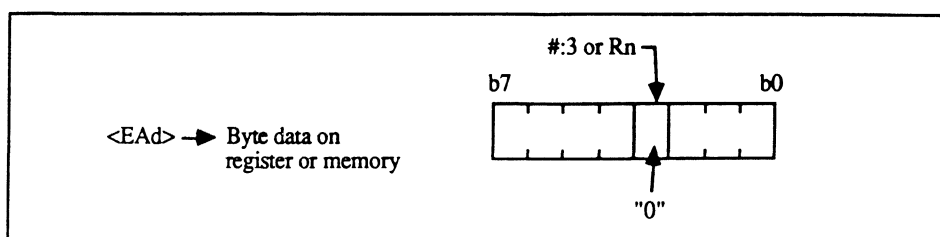


Figure A-4. BCLR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Operand Mnemonic	Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BCLR	#:3, Rd	7	2	0	Imm rd	2
Register indirect	BCLR	#:3, @Rd	7	D	0	rd 0 7 2 0	Imm 0 8
Absolute address	BCLR	#:3, @aaaa:8	7	F	abs	7 2 0	Imm 0 8
Register direct	BCLR	Rn, Rd	6	2	rn	rd	2
Register indirect	BCLR	Rn, @Rd	7	D	0	rd 0 6 2 rn	0 8
Absolute address	BCLR	Rn, @aaaa:8	7	F	abs	6 2 rn	0 8

Notes: "Addressing mode" above specifies the destination effective address <EAd>.

A.2.10 BIANd (Bit Invert AND)

Assembler Format: BIANd #:3, <EAd>

Operation: $C \wedge \sim(\text{Bit number of Destination}) \rightarrow C$

Description: Inverts a bit in destination operand, then performs a logical AND operation between the inverted bit and the C flag, and loads the result to the C flag (Figure A-5). The Bit number is specified by 3-bit immediate data. The destination operand is not affected by the BIANd instruction.

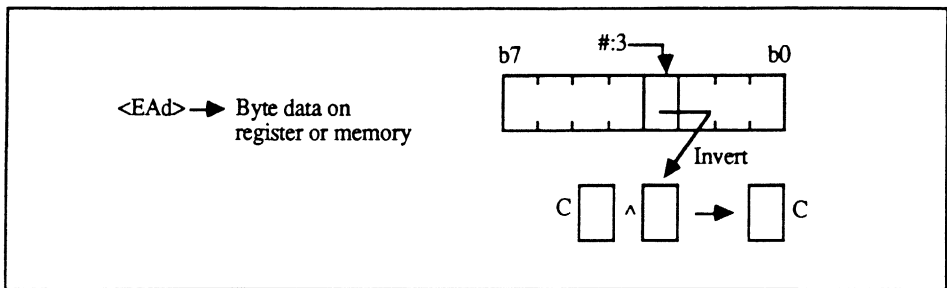


Figure A-5. BIANd Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: Set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format							Exec. States			
			1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte				
Register direct	BIAND	#:3, Rd	7	6	1	Imm	rd			2			
Register indirect	BIAND	#:3, @Rd	7	C	0	rd	0	7	6	1	Imm	0	6
Absolute address	BIAND	#:3, @aaa:8	7	E	abs			7	6	1	Imm	0	6

Notes: "Addressing mode" above specifies destination the effective address <EAd>.

A.2.11 BILD (Bit Invert Load)

Assembler Format: BILD #3, <EAd>

Operation: ~ (<Bit number > of <Destination>) -> C

Description: Inverts a bit of the destination operand, and loads the inverted bit to the C flag (Figure A-6). The Bit number is specified by 3-bit immediate data. The destination operand is not affected by the BILD instruction.

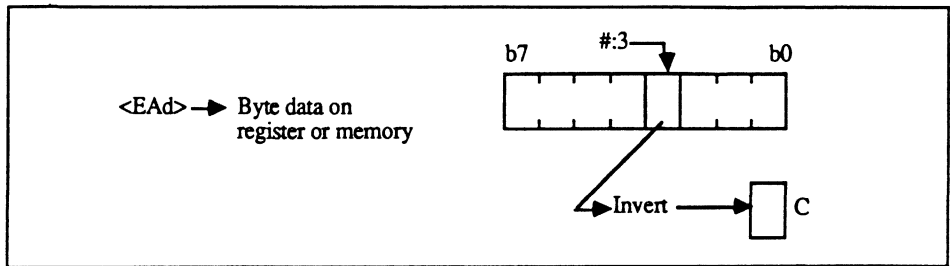


Figure A-6. BILD Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: Set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BILD	#:3, Rd	7	7	1 Imm rd		2
Register indirect	BILD	#:3, @Rd	7	C	0 rd 0	7 7 1 Imm 0	6
Absolute address	BILD	#:3, @aaaa:8	7	E	abs	7 7 1 Imm 0	6

Notes: "Addressing mode" above specifies the destination effective address <EAd>.

A.2.12 BIOR (Bit Invert OR)

Assembler Format: BIOR #3, <EAd>

Operation: C v ~(<Bit number > of <Destination>) -> C

Description: Inverts a bit in destination operand, then performs a logical OR operation between the inverted bit and the C flag, and loads the result to the C flag (Figure A-7). The Bit number is specified by 3-bit immediate data. The destination operand is not affected by the BIOR instruction.

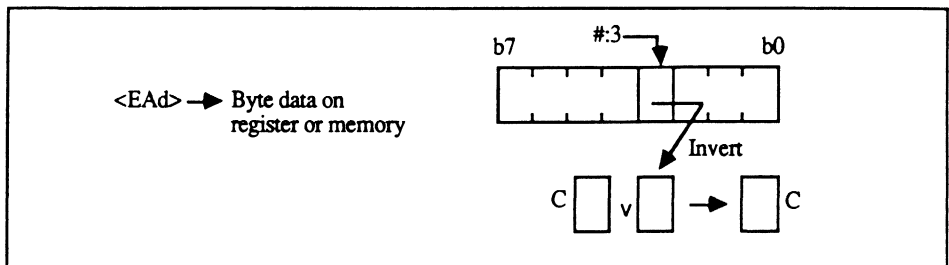


Figure A-7. BIOR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: Set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BIOR	#:3, Rd	7	4	1 Imm rd		2
Register indirect	BIOR	#:3, @Rd	7	C	0 rd 0	7 4 1 Imm 0	6
Absolute address	BIOR	#:3, @aaaa:	8	7	E abs	7 4 1 Imm 0	6

Notes: "Addressing mode" above specifies the destination effective address <EAd>.

A.2.13 BIST (Bit Invert Store)

Assembler Format: BIST #:3, <EAd>

Operation: ~C -> (<Bit number > of <Destination>)

Description: Inverts the C flag, and loads the inverted C flag into a bit in the destination operand (Figure A-8). The Bit number is specified by 3-bit immediate data. Bits other than the specified bit are not affected by the BIST instruction.

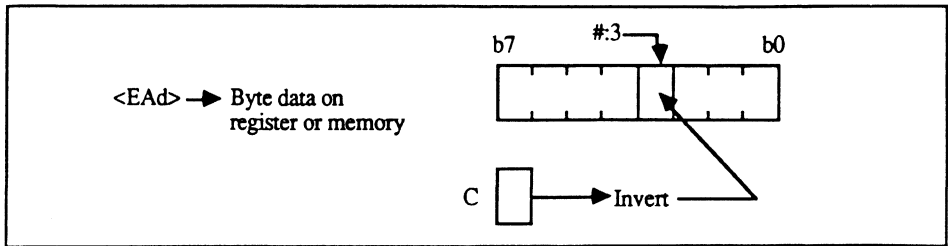


Figure A-8. BIST Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BIST	#:3, Rd	6	7	1 Imm rd		2
Register indirect	BIST	#:3, @Rd	7	D	0 rd 0	6 7 1 Imm 0	8
Absolute address	BIST	#:3, @aaaa:8	7	F	abs	6 7 1 Imm 0	8

Notes: "Addressing mode" above specifies the destination effective address <EAd>.

A.2.14 BIXOR (Bit Invert Exclusive OR)

Assembler Format: BIXOR #:3, <EAd>

Operation: C ++ [~(<Bit number > of <Destination>)] -> C

Description: Inverts a bit of the destination operand, then performs an exclusive OR operation between the inverted bit and the C flag, and loads the result into the C flag (Figure A-9). The Bit number is specified by 3-bit immediate data. The destination operand is not affected by the BIXOR instruction.

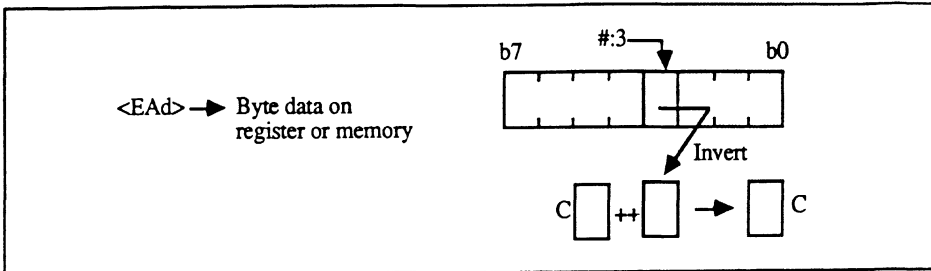


Figure A-9. BIOR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: Set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register direct	BIXOR	#:3, Rd	7	5	1	Imm rd	2
Register indirect	BIXOR	#:3, @Rd	7	C	0	rd 0 7 5 1	Imm 0 6
Absolute address	BIXOR	#:3, @aaaa:	8	7	E	abs 7 5 1	Imm 0 6

Notes: "Addressing mode" above specifies the destination effective address <EAd>.

A.2.15 BLD (Bit Load)

Assembler Format: BLD #:3, <EAd>

Operation: (<Bit number> of <Destination>) -> C

Description: Loads a bit of the destination operand to the C flag (Figure A-10). The bit number is specified by 3-bit immediate data. The destination operand is not affected.

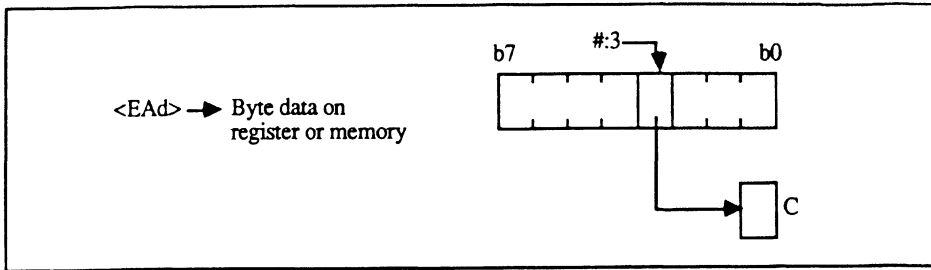


Figure A-10. BLD Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

*S: C set if a bit is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BLD	#:3, Rd	7	7	0 Imm rd		2
Register indirect	BLD	#:3, @Rd	7	C	0 rd 0	7 7 0 Imm 0	6
Absolute address	BLD	#:3, @aaaa:	8	7	E abs	7 7 0 Imm 0	6

Notes: "Addressing mode" above specifies the destination effective address <EAd>.

A.2.16 BNOT (Bit Not)

Assembler Format: BNOT #:3, <EAd>
 BNOT Rn, <EAd>

Operation: \sim (<Bit number> of <Destination>) -> (<Bit number> of <Destination>)

Description: Inverts a bit of the destination operand (Figure A-11). The bit number is specified by 3-bit immediate data or the lower 3 bits of a general purpose register. The bit is not tested and the CCR is not affected.

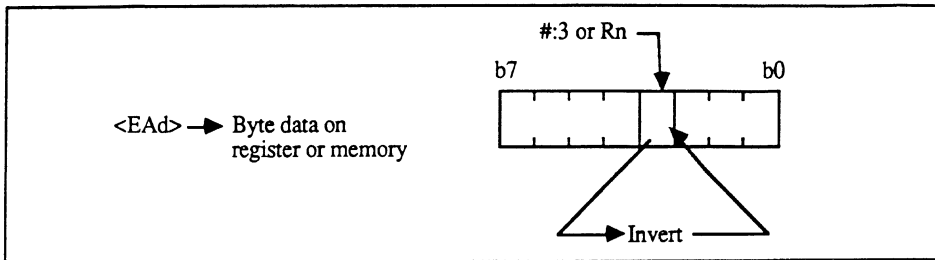


Figure A-11. BNOT Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format								Exec. States		
			1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte	8th Byte			
Register direct	BNOT	#:3, Rd	7	1	0	Imm	rd					2	
Register indirect	BNOT	#:3, @Rd	7	D	0	rd	0	7	1	0	Imm	0	8
Absolute address	BNOT	#:3, @aaaa:8	7	F	abs			7	1	0	Imm	0	8
Register direct	BNOT	Rn, Rd	6	1	m	rd							2
Register indirect	BNOT	Rn, @Rd	7	D	0	rd	0	6	1	m	0		8
Absolute address	BNOT	Rn, @aaaa:8	7	F	abs			6	1	m	0		8

Notes: "Addressing mode" above specifies the destination operand effective address <EAd>.

A.2.17 BOR (Bit Inclusive OR)

Assembler Format: BOR #:3, <EAd>

Operation: C v (<Bit number> of <Destination>) -> C

Description: Performs a logical OR operation between a bit of the destination operand and the C flag, and loads the result to the C flag (Figure A-12). The bit number is specified by a 3-bit immediate data. The destination operand is not affected by the BOR instruction.

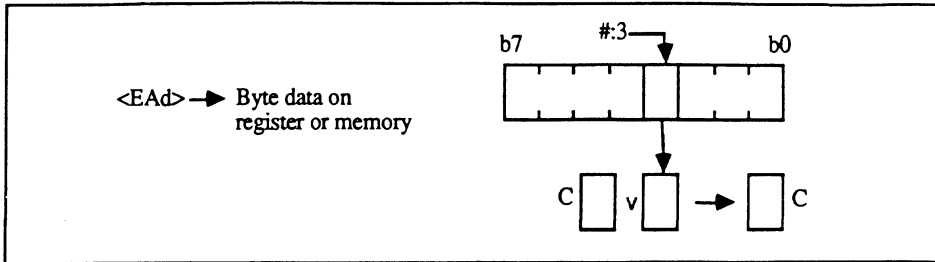


Figure A-12. BOR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: S set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

		Instruction Format											
Addressing Mode	Mnemonic	Operand Format	Operand								Exec. States		
			1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte	8th Byte			
Register direct	BOR	#:3, Rd	7	4	0	Imm	rd					2	
Register indirect	BOR	#:3, @Rd	7	C	0	rd	0	7	4	0	Imm	0	6
Absolute address	BOR	#:3, @aaaa:	8	7	E	abs		7	4	0	Imm	0	6

Notes: "Addressing mode" above specifies the destination operand effective address <EAd>.

A.2.18 BRA (Branch Always)

Assembler Format: BRA disp.

Operation: PC + disp -> PC

Description: Always branches to an address (branch destination address) gained by adding 7-bit signed displacement to the PC. The PC points to a next instruction address. Accordingly, the branch destination address can be within the range of -126 bytes to +128 bytes from the BRA instruction address.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
PC relative	BRA	disp	4	0	disp		4

Notes: Branch destination address must be an even address. The machine code of BRA is the same as that of BT (Branch if true).

A.2.19 BRN (Branch Never)

Assembler Format: BRN disp.

Operation: No operation is performed. Only the PC is incremented.

Description: Does not branch but executes the next sequential instruction. BRN can be replaced with two NOP instructions.

Operand Size: None

Condition Codes:

I - H - N Z V C
U U U U U U U U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
PC relative	BRN	disp	4	1	disp	4	

Notes: Machine code of BRN is the same as that of BF (Branch if false).

A.2.20 BSET (Bit Set)

Assembler Format: BSET #:3, <EAd>
BSET Rn, <EAd>

Operation: 1-> (<Bit number> of <Destination>)

Description: Sets a bit in destination operand (Figure A-13). The bit number is specified by a 3-bit immediate data. The bit is not tested and the CCR is not affected.

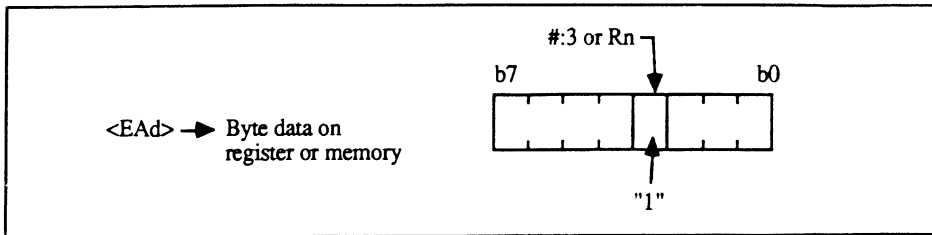


Figure A-13. BSET Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

		Instruction Format											
Addressing Mode	Mnemonic	Operand Format	Operand							Exec. States			
			1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte				
Register direct	BSET	#:3, Rd	7	0	0	Imm	rd			2			
Register indirect	BSET	#:3, @Rd	7	D	0	rd	0	7	0	0	Imm	0	8
Absolute address	BSET	#:3, @aaaa:8	7	F	abs			7	0	0	Imm	0	8
Register direct	BSET	Rn, Rd	6	0	m	rd							2
Register indirect	BSET	Rn, @Rd	7	D	0	rd	0	6	0	m	0	8	
Absolute address	BSET	Rn, @aaaa:8	7	F	abs			6	0	m	0	8	

Notes: "Addressing mode" above specifies the destination operand effective address <EAd>.

A.2.21 BSR (Branch to Subroutine)

Assembler Format: BSR disp.

Operation: PC -> @-SP
 PC + disp -> PC

Description: Pushes the current PC to the stack, and branches to address (branch destination address) gained by adding the 7-bit signed displacement to the PC. The PC points to a next instruction address. Accordingly, the branch destination address can be within the range of -126 bytes to +128 bytes from the BSR instruction address.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	1st	2nd	3rd	4th	Exec. States
			Byte	Byte	Byte	Byte	
PC relative	BSR	disp	5	5	disp		6

Notes: Branch destination address must be an even address.

A.2.22 BST (Bit Store)

Assembler Format: BST #3, <EAd>

Operation: C -> (<Bit number> of <Destination>)

Description: Stores content of the C flag to a bit of the destination operand (Figure A-14). The bit number is specified by 3-bit immediate data. Bits in the destination other than the specified bit, are not affected by the BST instruction.

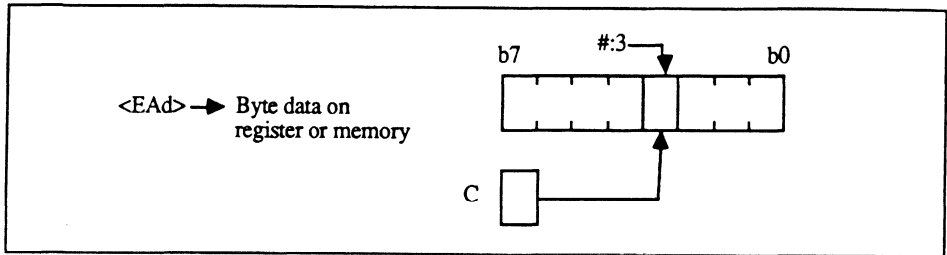


Figure A-14. BST Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BST	#:3, Rd	6	7	0 Imm rd		2
Register indirect	BST	#:3, @Rd	7	D	0 rd 0	6 7 0 Imm 0	8
Absolute address	BST	#:3, @aaaa:	8	7	F abs	6 7 0 Imm 0	8

Notes: "Addressing mode" above specifies the destination operand effective address <EAd>.

A.2.23 BTST (Bit Test)

Assembler Format: BTST #3, <EAd>
BTST Rn, <EAd>

Operation: ~(<Bit number> of <Destination>) -> Z

Description: Tests a bit of the destination operand and loads the result into the Z flag (Figure A-15). The bit number is specified by 3-bit immediate data or the lower 3 bits of a general purpose register. The destination operand is not affected by the BTST instruction.

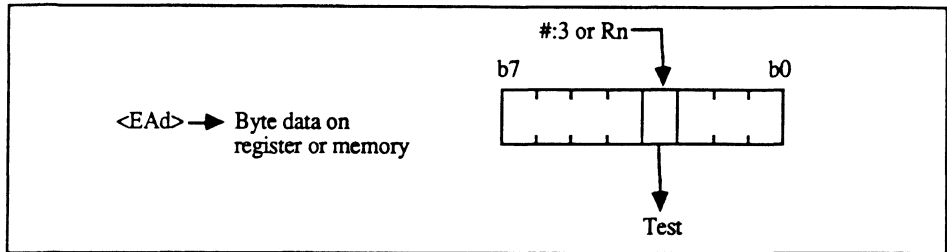


Figure A-15. BTST Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	S*	U	U

S*: Set if the tested bit is "0"; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format							Exec. States	
			1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte		
Register direct	BTST	#:3, Rd	7	3	0	Imm rd					2
Register indirect	BTST	#:3, @Rd	7	C	0	rd 0	7	3	0	Imm 0	6
Absolute address	BTST	#:3, @aaaa:	8	7	E	abs	7	3	0	Imm 0	6
Register direct	BTST	Rn, Rd	6	3	m	rd					2
Register indirect	BTST	Rn, @Rd	7	C	0	rd 0	6	3	m	0	6
Absolute address	BTST	Rn, @aaaa:	8	7	E	abs	6	3	m	0	6

Notes: "Addressing mode" above specifies the destination operand effective address <EAd>.

A.2.24 BXOR (Bit Exclusive OR)

Assembler Format: BXOR #:3, <EAd>

Operation: C ++ (<Bit number> of <Destination>) -> C

Description: Performs an exclusive OR operation between a bit of the destination operand and the C flag, and loads the result into the C flag (Figure A-16). The bit number is specified by 3-bit immediate data. The destination operand is not affected by the BXOR instruction.

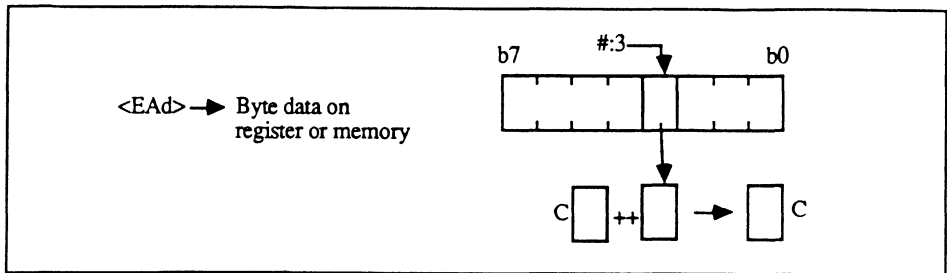


Figure A-16. BXOR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	S*

S*: S set if result is "1"; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	BXOR	#:3, Rd	7	5 0	Imm rd	.	2
Register indirect	BXOR	#:3, @Rd	7	C 0	rd 0	7 5 0	Imm 0 6
Absolute address	BXOR	#:3, @aaaa:	8 7	E	abs	7 5 0	Imm 0 6

Notes: "Addressing mode" above specifies the destination operand effective address <EAd>.

A.2.25 CMP (Compare)

Assembler Format: CMP.B <EAs>, Rd

Operation: (Destination) - (Source)

Description: Subtracts the source operand from the destination operand, and changes the CCR according to the result. The destination operand is not affected by the CMP instruction.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	-	S	S	S	S

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Immediate	CMP.B	#:8, Rd	A rd	Imm			2
Register direct	CMP.B	Rs, Rd	1 C	rs rd			2

Notes:

A.2.26 CMP (Compare)

Assembler Format: CMP.W Rs, Rd

Operation: (Destination) - (Source)

Description: Subtracts the source operand from the destination operand, and change the CCR according to the result. The destination operand is not affected by the CMP instruction.

Operand Size: Word

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S	S	S

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	CMP.W	Rs, Rd	1	D	0 rs	0 rd	2

Notes:

A.2.27 DAA (Decimal Adjust Add)

Assembler Format: DAA Rd

Operation: Decimal adjusted (Destination) -> (Destination)

Description: Performs decimal adjustment on the destination operand after BCD addition by the ADD.B or ADDX.B instruction. Decimal adjustment is performed by adding "00H", "06H", "60H", or "66H" according to the C, H flags and the destination operand value as shown below.

C Before DAA	Upper 4 Bits Before DAA	H Before DAA	Lower 4 Bits Before DAA	Data to be Added (Hex.)	C After DAA
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

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	*	U	S	S	*	S*

S*: sets to 1 if carry is generated from bit 7, unchanged otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	DAA	Rd	0 F	0 rd			2

Notes:

A.2.28 DAS (Decimal Adjust Subtract)

Assembler Format: DAS Rd

Operation: Decimal adjusted (Destination) -> (Destination)

Description: Performs decimal adjustment on the destination operand after BCD subtraction by the SUB.B or SUBX.B instruction. Decimal adjustment is performed by adding "00H", "FAH", "A0H", or "9AH" according to the C, H flags and the destination operand value as shown below.

C Before DAS	Upper 4 Bits Before DAS	H Before DAS	Lower 4 Bits Before DAS	Data to be Added (Hex.)	C After DAS
0	0-9	0	0-9	00	0
0	0-8	1	6-F	FA	0
1	7-F	0	0-9	A0	1
1	6-F	1	6-F	9A	1

Operand Size: Byte

Condition Codes:

I - H - N Z V C
 U U * U S U * U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	DAS	Rd	1 F	0 rd			2

Notes:

A.2.29 DEC (Decrement)

Assembler Format: DEC.B Rd

Operation: (Destination) -1 -> (Destination)

Description: Subtracts "1" from destination operand and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	S*	U

*S: V set if result overflows (if destination operand before DEC is "80H"); cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	DEC.B	Rd	1	A 0	rd	2	

Notes:

A.2.30 DIVXU (Divide Extended as Unsigned)

Assembler Format: DIVXU Rs, Rd

Operation: (Destination) + (Source) -> (Destination)

Description: Divides 16-bit destination operand by 8-bit source operand, and loads the result to the destination. A 16-bit register and 8-bit register must be specified as destination and source, respectively. After division, the lower byte of the destination contains an 8-bit quotient and the upper byte of the destination an 8-bit remainder. See Figure A-17.

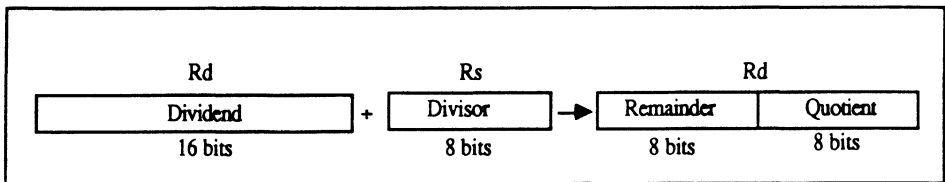


Figure A-17. DIVXU Operation

Note that the result cannot be guaranteed if division by zero is performed or if the result overflows. Since the DIVXU instruction performs the operation: "16-bit dividend + 8-bit divisor -> 8-bit quotient and 8-bit remainder", the result might overflow. However, an overflow can be avoided by using the following routine, which requires, however, some work registers.

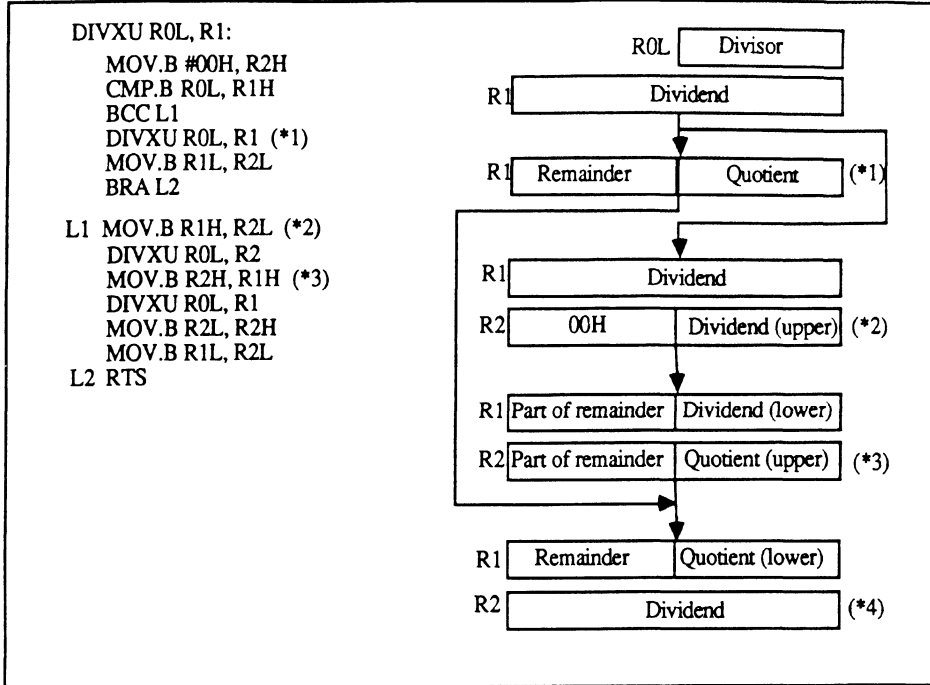


Figure A-18. Overflow Prevention Program

Operand Size: Byte

Condition Codes:

I - H - N Z V C

U U U U S* U U

S*: Z set if divisor is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	DIVXU	Rs, Rd	5	1	rs 0 rd	14	

Notes:

A.2.31 EEPMOV (Move Data to EEPROM)

Assembler Format: EEPMOV

Operation: Repeat @R5+ -> @R6+

R4L -1 -> R4L

Until R4L = 0

Description: EEPMOV is a kind of block data transfer instruction. Data in memory, whose start address is specified by R5 and whose byte size is specified by R4L, is loaded to memory whose start address is specified by R6. R4L is decremented after one byte data transfer. This operation continues until R4L reaches 0. Detection of interrupt requests is not done during data transfer. After the EEPMOV instruction, R4L is 00H, and R5 and R6 contain the values of the last address + 1.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format							Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	5th Byte	6th Byte	7th Byte	
-	EEPMOV	7 B	5	C	5	9	8	F	*	

*: Number of execution states varies depending on the number of transfer data.

Notes:

A.2.32 INC (Increment)

Assembler Format: ICN.B Rd

Operation: (Destination) + 1 -> (Destination)

Description: Increments the destination operand by "1" and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C	S*	sets to 1 if destination operand before INC is 7FH, cleared otherwise.
U	U	U	U	S	S	S*	U		

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	INC.B	Rd	0 A	0 rd			2

Notes:

A.2.33 JMP (Jump)

Assembler Format: JMP <EAd>

Operation: (Destination) -> PC

Description: Jumps to the specified address (jump destination address).

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register indirect	JMP	@Rd	5	9 0 m	0		4
Absolute address	JMP	@@aaaa:16	5	A	0 0	abs	6
Memory indirect	JMP	@@aaaa:8	5	B	abs		8

Notes:

A.2.34 JSR (Jump to Subroutine)

Assembler Format: JSR <EA>

Operation: (Destination) -> PC

Description: Pushes the current PC onto the stack, and jumps to the specified address (jump destination address). The stacked PC contains the next instruction address.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register indirect	JSR	@Rd	5	D 0 m 0			6
Absolute address	JSR	@@aaaa:16	5	E 0 0		abs	8
Memory indirect	JSR	@@aaaa:8	5	F abs			8

Notes:

A.2.35 LDC (Load to Control Register)

Assembler Format: LDC <EAs>, CCR

Operation: (Source) -> CCR

Description: Loads the source operand to the CCR. The bits 6 and 4 in the CCR can be handled in the same way as the other bits in the CCR.

Note that no interrupts including NMI can be detected just after the LDC instruction cycle.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
S*	S*	S*	S*	S*	S*	S*	S*

S*: Same as the corresponding bit of the source operand

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Immediate	LDC	#:8,CCR	0	7	Imm		2
Register direct	LDC	Rs,CCR	0	3	0	rs	2

Notes:

A.2.36 MOV (Move Data)

Assembler Format: MOV.B Rs, Rd

Operation: (Source) -> (Destination)

Description: Moves the source operand to the destination. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S*	S*	0	U

S*:N set if the source operand is negative; cleared otherwise.

Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	MOV.B	Rs, Rd	0	C	rs	rd	2

Notes:

A.2.37 MOV (Move Data)

Assembler Format: MOV.W Rs, Rd

Operation: (Source) -> (Destination)

Description: Moves the source operand to the destination. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Word

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S*	S*	0	U

S*: N set if the source operand is negative; cleared otherwise.

Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register direct	MOV.W	Rs, Rd	0	D	0	rs	0 rd 2

Notes:

A.2.38 MOV (Move Data)

Assembler Format: MOV.B <EAs>, Rd

Operation: (Source) -> (Destination)

Description: Moves the source operand to the destination. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S*	S*	0	U

S*: N set if the source operand is negative; cleared otherwise.

Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States	
			1st Byte	2nd Byte	3rd Byte	4th Byte		
Immediate	MOV.B	#:8, Rd	F	rd	Imm		2	
Register indirect	MOV.B	@Rs, Rd	6	8	0 rs	rd	4	
Register indirect with displacement	MOV.B	@(disp:16, Rs), Rd	6	E	0 rs	rd	disp	6
Register indirect with post-increment	MOV.B	@Rs+, Rd	6	C	0 rs	rd		6
Absolute address	MOV.B	@aaaa:8, Rd	2	rd	abs			4
Absolute address	MOV.B	@aaaa:16, Rd	6	A	0	rd	abs	6

Notes: MOV.B @R7+, Rd must not be used since data in R7 becomes odd. See "5.4 Stack Notes" for details.

A.2.39 MOV (Move Data)

Assembler Format: MOV.W <EAs>, Rd

Operation: (Source) -> (Destination)

Description: Moves the source operand to the destination. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Word

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S*	S*	0	U

S*: N set if the source operand is negative; cleared otherwise.
 Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States	
Immediate	MOV.W	#:16, Rd	7	9	0	0 rd	Imm	4
Register indirect	MOV.W	@Rs, Rd	6	9	0	rs 0 rd		4
Register indirect with displacement	MOV.W	@(disp:16, Rs),Rd	6	F	0	rs 0 rd	disp	6
Register indirect with post-increment	MOV.W	@Rs+, Rd	6	D	0	rs 0 rd		6
Absolute address	MOV.W	@aaaa:16, Rd	6	B	0	0 rd	abs	6

Notes: Address <EAs> must be even. MOV.W @R7+, Rd is used to restore 16-bit register Rd from the stack.

A.2.40 MOV (Move Data)

Assembler Format: MOV.B Rs, <EAd>

Operation: (Source) -> (Destination)

Description: Moves the source operand to the destination. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Byte

Condition Codes:

I - H - N Z V C
 U U U U S* S* 0 U

S*: N set if the source operand is negative; cleared otherwise.

Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register indirect	MOV.B	Rs, @Rd	6 8	1 rd	rs		4
Register indirect with displacement	MOV.B	Rs, @(disp:16, Rd)	6 E	1 rd	rs	disp	6
Register indirect with pre-decrement	MOV.B	Rs, @-Rd	6 C	1 rd	rs		6
Absolute address	MOV.B	Rs, @aaaa:8	3	rs	abs		4
Absolute address	MOV.B	Rs, @aaaa:16	6 A	8	rs	abs	6

Notes: MOV.B Rs, @-R7 must not be used since data in R7 becomes odd. See "5.6 Stack Note" for details.

A.2.41 MOV (Move Data)

Assembler Format: MOV.W Rs, <EAd>

Operation: (Source) -> (Destination)

Description: Moves the source operand to the destination. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Word

Condition Codes:

I - H - N Z V C

U U U U S* S* 0 U

S*: N set if the source operand is negative; cleared otherwise.

Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register indirect	MOV.W	Rs, @Rd	6	9	1 rd 0 rs		4
Register indirect with displacement	MOV.W	Rs, @(disp:16, Rd)	6	F	1 rd 0 rs	disp	6
Register indirect with pre-decrement	MOV.W	Rs, @-Rd	6	D	1 rd 0 rs		6
Absolute address	MOV.W	Rs, @aaaa:16	6	B	8 0 rs	abs	6

Notes: Address <EAd> must be even. MOV.W Rs, @-R7 is used to push 16-bit register Rs onto the stack.

A.2.42 MOVFPE (Move Data from Peripheral with E Clock)

Assembler Format: MOVFPE @aaaa:16,Rd

Operation: (Source) -> (Register)

Description: Moves the source operand in memory specified by 16-bit absolute address to destination synchronously with E clock. The source operand is tested and the CCR is changed according to the check result.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S*	S*	0	U

S*: N set if the source operand is negative; cleared otherwise.

Z set if the source operand is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States	
Absolute address	MOVFPE	@aaaa:16, Rd	6	A	4	rd	abs	*

*: Undefined.

Notes: MOVFPE can only use absolute addressing mode and only byte data. The number of MOVFPE execution cycles is undefined. MOVFPE requires at least 9-16 states for a data transfer, which varies depending on what state E clock gets in when MOVFPE starts to execute.

A.2.43 MOVTPPE (Move Data to Peripheral with E Clock)

Assembler Format: MOVTPPE Rs, @aaaa:16

Operation: (Register) -> (Destination)

Description: Moves data from a register to destination specified by 16-bit absolute address synchronously with E clock. The data in register is tested and the CCR is changed according to the check result.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S*	S*	0	U

S*: N set if data in register is negative; cleared otherwise.

Z set if data in register is zero; cleared otherwise.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States	
Absolute address	MOVTPPE	Rs, @aaaa:16	6	A	C	rs	abs	*

*: Undefined.

Notes: MOVTPPE can only use absolute addressing mode and only byte data. The number of MOVTPPE execution cycles is undefined. MOVTPPE requires at least 9-16 states for a data transfer, which varies depending on what state E clock gets in when MOVTPPE starts to execute.

A.2.44 MULXU (Multiply Extended as Unsigned)

Assembler Format: MULXU Rs, Rd

Operation: (Destination) x (Source) -> (Destination)

Description: Multiplies 8-bit destination operand with 8-bit source operand, and loads 16-bit result into the destination (Figure A-19). Accordingly, an 8-bit register and 16-bit register must be specified as source and destination, respectively. The upper 8 bits of the 16-bit destination register can also be used as the source.

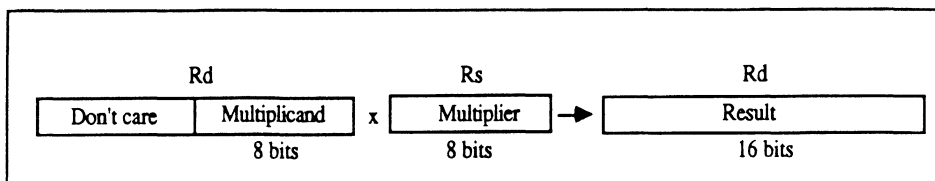


Figure A-19. MULXU Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register direct	MULXU	Rs, Rd	5	0	rs	0 rd	14

Notes:

A.2.45 NEG (Negate)

Assembler Format: NEG.B Rd

Operation: 0 - (Destination) -> (Destination)

Description: Takes two's complement of destination operand by subtracting the destination operand from 00H, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S	S	S

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	NEG.B	Rd	1 7	8 rd		2	

Notes:

A.2.46 NOP (No Operation)

Assembler Format: NOP

Operation: No operation is performed.

Description: No operation is performed. The CPU internal status is not affected.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
-	NOP		0	0	0	0	2

Notes:

A.2.47 NOT (Logical Complement)

Assembler Format: NOT.B Rd

Operation: ~(Destination) -> (Destination)

Description: Take one's complement of destination operand by subtracting the destination operand from FFH, and load the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	U

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	NOT.B	Rd	1 7	0 rd		2	

Notes:

A.2.48 OR (Inclusive OR Logical)

Assembler Format: OR.B <EAs>, Rd

Operation: (Destination) v (Source) -> (Destination)

Description: Performs a logical OR operation between the destination operand and the source operand, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	OR.B	#:8, Rd	C rd	Imm			2
Register direct	OR.B	Rs, Rd	1 4	rs rd			2

Notes:

A.2.49 ORC (Inclusive OR Control Register)

Assembler Format: ORC #:8, CCR

Operation: CCR v Immediate Data -> CCR

Description: Performs a logical OR operation between the CCR and immediate data, and loads the result to the CCR. Bits 6 and 4 of the CCR can be handled in the same way as the other bits of the CCR. Note that no interrupts including NMI can be detected just after the ORC instruction cycle.

Operand Size: Byte

Condition Codes:

I - H - N Z V C

S* S* S* S* S* S* S* S*

S*: Same as the corresponding bit of the immediate data.

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	ORC	#:8, CCR	0	4	Imm	2	

Notes:

A.2.50 ROTXL (Rotate Left with Extended Carry)

Assembler Format: ROTXL.B Rd

Operation: (Destination) rotate -> (Destination)

Description: Rotates left destination operand with the C flag. The MSB of destination operand is copied to the C flag and the C flag is copied to the LSB. The rotate count is fixed to 1.

See Figure A-20.

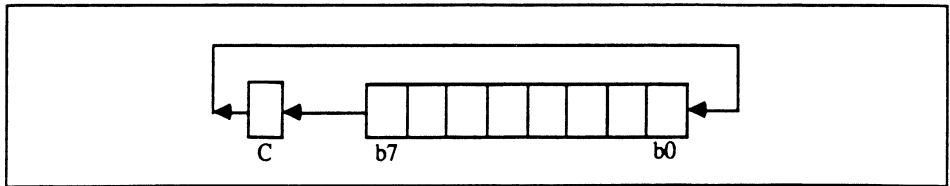


Figure A-20. ROTXL Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	S*

S*: C same as bit 7 of destination operand prior to operation.

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	ROTXL.B	Rd	1	2	0	rd	2

Notes:

A.2.51 ROTXR (Rotate Right with Extended Carry)

Assembler Format: ROTXR.B Rd

Operation: (Destination) rotate -> (Destination)

Description: Rotates right destination operand with the C flag. The LSB of destination operand is copied to the C flag and the C flag is copied to the MSB. The rotate count is fixed to 1. See Figure A-21.

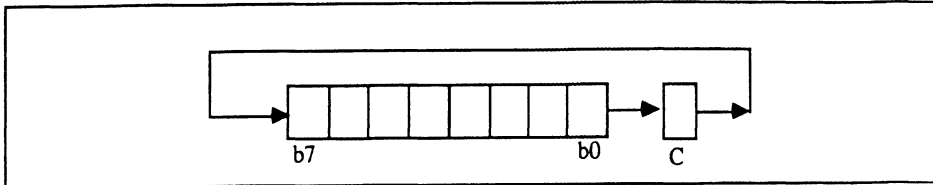


Figure A-21. ROTXR Operation

Operand Size: Byte

Condition Codes:

I - H - N Z V C

U U U U S S 0 S*

S*: C same as bit 0 of destination operand prior to operation.

Operand Format and Number of Execution States:

Addressing Mode	Operand Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	ROTXR.B	Rd	1	3	0	rd	2

Notes:

A.2.52 ROTL (Rotate Left)

Assembler Format: ROTL.B Rd

Operation: (Destination) rotate -> (Destination)

Description: Rotates left the destination operand. The MSB of destination operand is copied to both the C flag and LSB. The rotate count is fixed to 1. See Figure A-22.

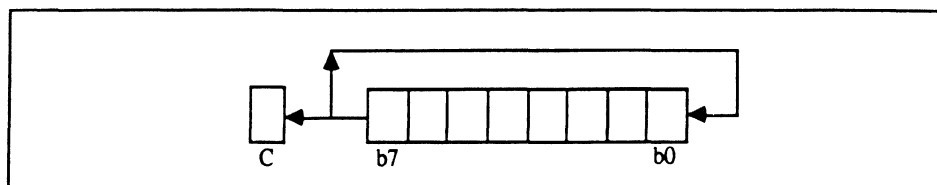


Figure A-22. ROTL Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	S*

S*: C same as bit 7 of destination operand prior to operation.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register direct	ROTL.B	Rd	1	2	8	rd	2

Notes:

A.2.53 ROTR (Rotate Right)

Assembler Format: ROTR.B Rd

Operation: (Destination) rotate -> (Destination)

Description: Rotates right the destination operand. The LSB of destination operand is copied to both the C flag and MSB. The rotate count is fixed to 1. See Figure A-23.

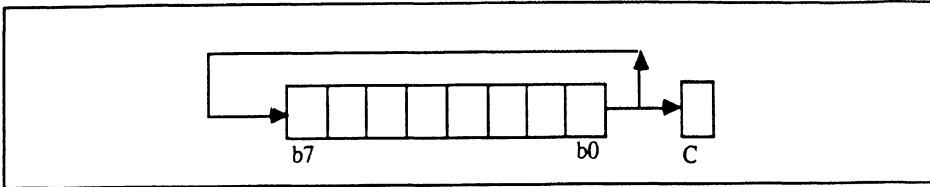


Figure A-23. ROTR Operation

Operand Size: Byte

Condition Codes:

I - H - N Z V C

U U U U S S 0 S*

S*: C same as bit 0 of destination operand prior to operation.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	ROTR.B	Rd	1	3	8	rd	2

Notes:

A.2.54 RTE (Return from Exception)

Assembler Format: RTE

Operation: @SP+ -> CCR
 @SP+ -> PC

Description: Restores the CCR and PC from the stack and begins processing from the address pointed to by the restored PC. The PC and CCR before the RTE instruction execution are replaced with the restored PC and CCR.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
S*	S*	S*	S*	S*	S*	S*	S*

S*: Same as the corresponding bit of the restored CCR from stack

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
-	RTE		5	6	7	0	10

Notes: The CCR, which is a byte size register, is restored as a word size register from the stack. At this time, the upper 8 bits are ignored. Accordingly, the SP is increment by 4 after RTE instruction execution since a word size CCR and a word size PC are restored.

A.2.55 RTS (Return from Subroutine)

Assembler Format: RTS

Operation: @SP+ -> PC

Description: Restores the PC from the stack and begins processing from the address pointed to by the restored PC. Note that the PC before the RTS execution is replaced with the restored PC.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
-	RTS		5	4	7	0	8

Notes:

A.2.56 SHAL (Shift Left Arithmetically)

Assembler Format: SHAL.B Rd

Operation: (Destination) shift -> (Destination)

Description: Shifts the destination operand to the left. The MSB of destination operand is copied to the C flag and "0" is loaded to the LSB. The shift count is fixed to 1. See Figure A-24.

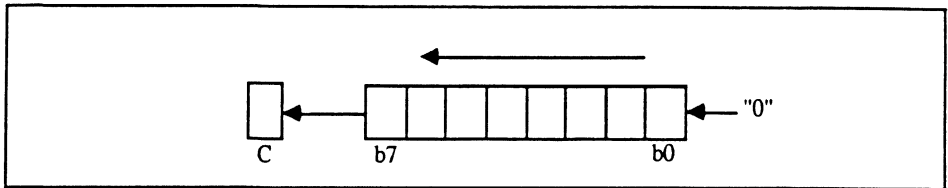


Figure A-24. SHAL Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	S*

S*: C same as bit 7 of destination operand prior to operation.

Operand Format and Number of Execution States:

		Instruction Format					
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States
Register direct	SHAL.B	Rd	1	0	8	rd	2

Notes:

A.2.57 SHAR (Shift Right Arithmetically)

Assembler Format: SHAR.B Rd

Operation: (Destination) shift -> (Destination)

Description: Shifts the destination operand to the right. The MSB of destination operand keeps unchanged. The LSB is copied to the C flag. The shift count is fixed to 1. See Figure A-25.

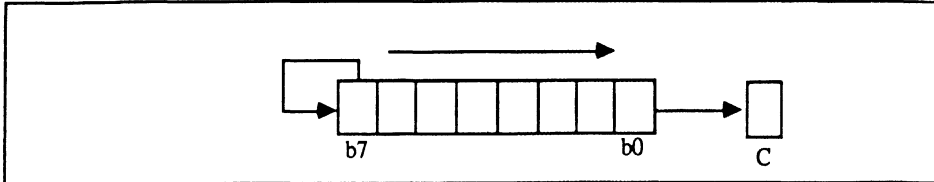


Figure A-25. SHAR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	S*

S*: C same as bit 0 of destination operand prior to operation.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	SHAR.B	Rd	1	1	8	rd	2

Notes:

A.2.58 SHLL (Shift Left Logically)

Assembler Format: SHLL.B Rd

Operation: (Destination) shift -> (Destination)

Description: Shifts the destination operand to the left. The MSB of destination operand is copied to the C flag and "0" is loaded to the MSB. The shift count is fixed to 1. See Figure A-26.

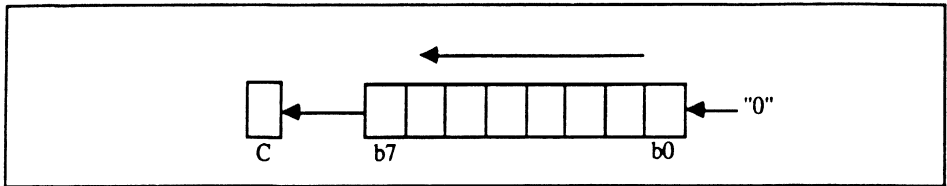


Figure A-26. SHLL Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	S*

S*: C same as bit 7 of destination operand prior to operation.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	SHLL.B	Rd	1	0	0	rd	2

Notes:

A.2.59 SHLR (Shift Right logically)

Assembler Format: SHLR.B Rd

Operation: (Destination) shift -> (Destination)

Description: Shifts the destination operand to the right. The LSB of destination operand is copied to the C flag and "0" is loaded to the MSB. The shift count is fixed to 1. See Figure A-27.

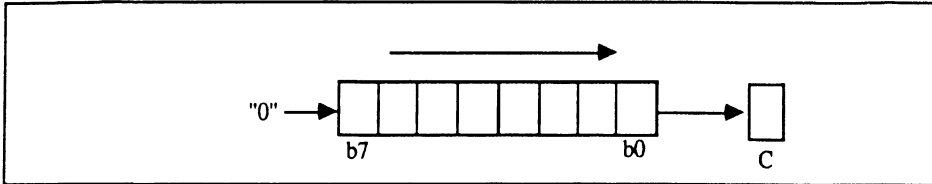


Figure A-27. SHLR Operation

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	S*

S*: C same as bit 0 of destination operand prior to operation.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	SHLR.B	Rd	1	1	0	rd	2

Notes:

A.2.60 SLEEP (Sleep)

Assembler Format: SLEEP

Operation: Operation mode -> Sleep mode

Description: The H8/300 enters sleep mode, in which an instruction execution is halted and the CPU internal status is retained. The CPU waits for an exception occurrence. If an exception occurs, sleep mode is cancelled and the CPU begins the corresponding exception processing. However, if the exception is masked, sleep mode cannot be cancelled.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
-	SLEEP		0	1	8	0	2

Notes: If the SLEEP instruction is executed when the SSBY (Software standby) bit of the SBYCR (Standby Control Register) register is set, the H8/300 enters software standby mode. See "6. Low Power Consumption Mode" for details.

A.2.61 STC (Store Control Register)

Assembler Format: STC CCR, Rd

Operation: CCR -> (Destination)

Description: Stores the condition code register (CCR) in the destination location. The bits 6 and 5 of the CCR can be stored in the same way as the other bits of the CCR.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	STC	CCR, Rd	0 2	0 rd			2

Notes:

A.2.62 SUB (Subtract Binary)

Assembler Format: SUB.B Rs, Rd

Operation: (Destination) - (Source) -> (Destination)

Description: Subtracts the source operand from the destination operand, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S	S	S

Operand Format and Number of Execution States:

Instruction Format							
Addressing Mode	Mnemonic	Operand Format	Operand				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	SUB.B	Rs, Rd	1	8	rs	rd	2

Notes:

A.2.63 SUB (Subtract Binary)

Assembler Format: SUB.W Rs, Rd

Operation: (Destination) - (Source) -> (Destination)

Description: Subtracts the source operand from the destination operand, and loads the result to the destination.

Operand Size: None

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S	S	S

Operand Format and Number of Execution States:

Instruction Format								
Addressing Mode	Mnemonic	Operand Format	1st Byte	2nd Byte	3rd Byte	4th Byte	Exec. States	
register direct	SUB.W	Rs, Rd	1	9	0 rs	0 rd	2	

Notes:

A.2.64 SUBX (Subtract with Extend Carry)

Assembler Format: SUBX.B <EAs>, Rd

Operation: (Destination) - (Source) - C -> (Destination)

Description: Subtracts the source operand and the C flag from the destination operand, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	S	U	S	S*	S	S

S* unchanged if the result is 0,
otherwise cleared to 0.

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	SUBX.B	#:8, Rd	B	rd	Imm		2
Register direct	SUBX.B	Rs, Rd	1	E	rs	rd	2

Notes:

A.2.65 SUBS (Subtract with Sign Extension)

Assembler Format: SUBS.W #1, Rd
SUBS.W #2, Rd

Operation: (Destination) - 1 -> (Destination)
(Destination) - 2 -> (Destination)

Description: Subtracts immediate data "1" or "2" from the destination operand, and loads the result to the destination. The CCR is not affected by the SUBS instruction.

Operand Size: Word

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	U	U	U	U

Operand Format and Number of Execution States:

Instruction Format

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Register direct	SUBS.W	#1, Rd	1 B	0 0 rd			2
Register direct	SUBS.W	#2, Rd	1 B	8 0 rd			2

Notes: The SUBS instruction cannot handle byte data.

A.2.66 XOR (Exclusive OR Logical)

Assembler Format: XOR.B <EAs>, Rd

Operation: (Destination) ++ (Source) -> (Destination)

Description: Performs an exclusive OR operation between the source operand and the destination operand, and loads the result to the destination.

Operand Size: Byte

Condition Codes:

I	-	H	-	N	Z	V	C
U	U	U	U	S	S	0	U

Operand Format and Number of Execution States:

Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	XOR.B	#:8, Rd	D rd	Imm			2
Register direct	XOR.B	Rs, Rd	1	5 rs rd			2

Notes:

A.2.67 XORC (Exclusive OR Control Register)

Assembler Format: XORC #:8, CCR

Operation: CCR ++ Immediate data -> CCR

Description: Performs an exclusive OR operation between the CCR and immediate data, and loads the result to the CCR. The bits 6 and 4 of the CCR can be handled in the same way as the other bits of the CCR. No interrupt including the NMI can be detected just after the XORC instruction cycle.

Operand Size: Byte

Condition Codes:

I - H - N Z V C
S* S* S* S* S* S* S* S*

S*: Same as the corresponding bit of the result.

Operand Format and Number of Execution States:

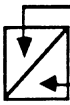
Addressing Mode	Mnemonic	Operand Format	Instruction Format				Exec. States
			1st Byte	2nd Byte	3rd Byte	4th Byte	
Immediate	XORC	#:8, CCR	0	5	Imm	2	

Notes:

Appendix. B Opcode Map *1

H L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	SHLL *2 SHAL			BRA	MULXU		BSET								
1	SLEEP	SHLR *2 SHAR			BRN	DIVXU		BNOT								
2	STC	ROTXL *2 ROTL			BHI			BCLR								
3	LDC	ROTXR *2 ROTR			BLS			BTST								
4	ORC	OR			BCC	RTS		BOR *2 BIOR BXOR *2 BIXOR BAND *2 BIAND								
5	XORC	XOR			BCS	BSR										
6	ANDC	AND			BNE	RTE										
7	LDC	NOT *2 NEG			BEQ			BST *2 BIST *2 BLD *2 BILD *2								
8					BVC											
9	ADD	SUB	MOV		BVS	JMP	MOV	ADD	ADDX	CMP	SUBX	OR	XOR	AND	MOV	
A	INC	DEC			BPL		*3									
B	ADDS	SUBS			BMI			EEMOV								
C					BGE											
D	MOV	CMP			BLT											
E	ADDX	SUBX			BGT	JSR		Bit inst.								
F	DAA	DAS			BLE											

Notes: 1. The above opcode map indicates only the first byte (Bits 15-8 of the first word) of each instruction code.

2.  When the MSB of the second byte (bit 7 of the first word) of the instruction code is "0".
- When the MSB of the second byte (bit 7 of the first word) of the instruction code is "1".

3. The MSB of the first and second bytes (bits 15 through bits 7 of the first word) of the MOVFPE, MOVTPPE are same as those of MOV instructions.

Appendix C. Instruction Set Summary

Symbols: O: 2-byte instruction

Δ: 4-byte instruction

B: Byte size

W: Word size

#xxx: Immediate

R: Register direct

@R: Register indirect

@(disp,R): Register indirect with displacement

@-R/@R+: Register indirect with pre-decrement/ register indirect with post-increment

@aaaa: Absolute address

@(disp,PC): Program counter relative

I: Interrupt mask bit

H: Half carry flag

N: Negative flag

Z: Zero flag

V: Overflow flag

C: Carry flag

PC: Program counter

CCR: Condition code register

SP: Stack pointer

Rs8/16, Rd8/16, Rn8/16: 8 or 16-bit general purpose register

#:3/8: 3 or 8-bit immediate data

disp:8/16: 8 or 16-bit displacement

aaaa: 8/16: 8 or 16-bit absolute address

->: Transmission direction

+: Addition

-: Subtraction

x: Multiplication

÷: Division

∧: Logical AND

∨: Logical OR

