To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

To all our customers

Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Software Manual

16

7900 Series

Software Manual

MITSUBISHI 16-BIT SINGLE-CHIP
MICROCOMPUTER
7700 Family / 7900 Series

Renesas Electronics
www.renesas.com

New publication, 1997.07

keep safety first in your circuit designs !

- Mitsubishi Electric Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Mitsubishi semiconductor product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Mitsubishi Electric Corporation or a third party.

- Mitsubishi Electric Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts or circuit application examples contained in these materials.

- All information contained in these materials, including product data, diagrams and charts, represent information on products at the time of publication of these materials, and are subject to change by Mitsubishi Electric Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for the latest product information before purchasing a product listed herein.

- Mitsubishi Electric Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.

- The prior written approval of Mitsubishi Electric Corporation is necessary to reprint or reproduce in whole or in part these materials.

- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of JAPAN and/or the country of destination is prohibited.

- Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for further details on these materials or the products contained therein.

REVISION DESCRIPTION LIST	7900 Series Software Manual

Rev. No.	Revision Description	Rev. date
1.0	First Edition	980731

Preface

This manual describes the software of the Mitsubishi CMOS 16-bit microcomputers, the 7900 Series. After reading this manual, the users will be able to understand the instruction set and the features about software of the 7900 Series, so that they can utilize their capabilities fully.

Table of contents

CHAPTER 1
DESCRIPTION

DESCRIPTION

The 7900 Series is upper compatible with the conventional 7700 Family.
The following outlines the features of the 7900 Series:

- Source-level-compatible with the conventional 7700 Family. (e.g., 7700 and 7751 Series).
- Whereas the 7700 and 7751 Series respectively support 103 and 109 instructions, the 7900 Series has its instruction set expanded to 203 instructions. The following instructions have been added:
  - (i) 32-bit operation instructions
  - (ii) 8-bit-data-dedicated instructions
  - (iii) Memory-to-memory data transfer instructions
  - (iv) Zero-clear instructions for register and memory
  - (v) Add/Subtract without-carry instructions
  - (vi) Add/Subtract instructions for stack pointer
  - (vii) OR, AND, and EOR instructions for memory
  - (viii) Compare instructions for memory
  - (ix) Signed conditional branch instructions
  - (x) Compare & Conditional branch instructions
  - (xi) Decrement & Conditional branch instructions
  - (xii) PC relative subroutine call instructions

Thanks to its expanded instruction set, the 7900 Series allows program sizes to be reduced by 20 to 30% on the average from the conventional 7700 Family.

- 16 Mbytes of memory space. Various addressing modes for accessing this memory space are available.
- A 64-Kbyte space from 000000_{16} to $00FFFF_{16}$ can be accessed at high speed by an instruction which has a small number of bytes. The 7900 Series has 4 direct page registers that can be used for this purpose.
- Reduced instruction execution cycles than the conventional 7700 Family.

CHAPTER 2

CENTRAL PROCESSING UNIT (CPU)

2.1 Central processing unit

The CPU (Central Processing Unit) has 13 registers as shown in Figure 2.1.1.

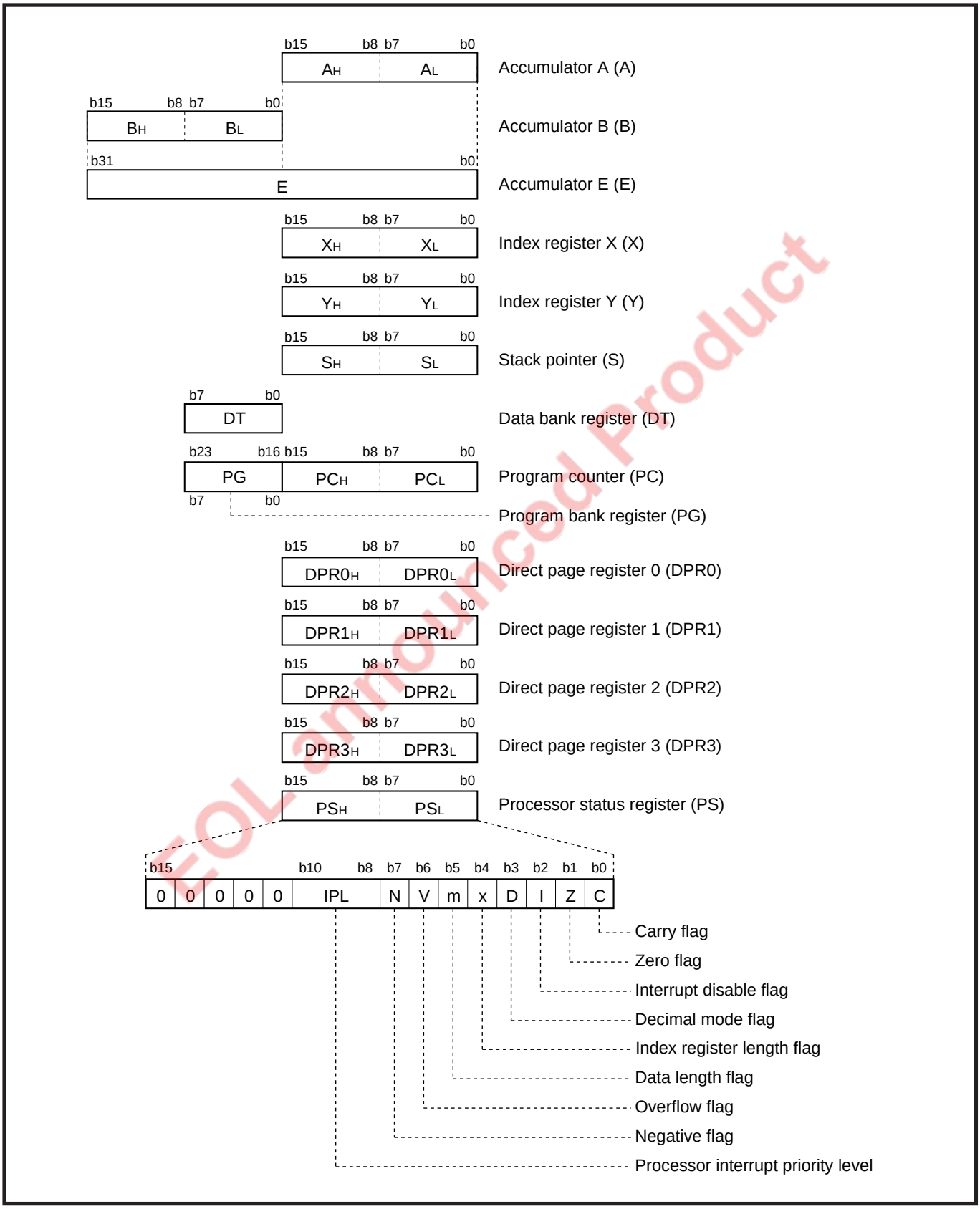


Fig. 2.1.1 CPU registers structure

2.1.1 Accumulator (Acc)

Accumulators A and B are available. Also, accumulators A and B can be connected in series for use as a 32-bit accumulator (accumulator E).

(1) Accumulator A (A)

Accumulator A is the main register of the microcomputer. The transaction of data such as calculation, data transfer, and input/output are performed mainly through accumulator A. It consists of 16 bits, and the low-order 8 bits can also be used separately. The data length flag (m) determines whether the register is used as a 16-bit register or as an 8-bit register. Flag m is a part of the processor status register which is described later. When an 8-bit register is selected, only the low-order 8 bits of accumulator A are used and the contents of the high-order 8 bits is unchanged.

(2) Accumulator B (B)

Accumulator B is a 16-bit register with the same function as accumulator A. Accumulator B can be used instead of accumulator A. The use of accumulator B, however except for some instructions, requires more instruction bytes and execution cycles than that of accumulator A. Accumulator B is also controlled by the data length flag (m) just as in accumulator A.

(3) Accumulator E (E)

This 32-bit accumulator consists of accumulator A for low-order 16 bits and accumulator B for high-order 16 bits. This accumulator is used for instructions that handle 32-bit data. It is not controlled by flag m.

2.1.2 Index register X (X)

Index register X consists of 16 bits and the low-order 8 bits can also be used separately. The index register length flag (x) determines whether the register is used as a 16-bit register or as an 8-bit register. Flag x is a part of the processor status register which is described later. When an 8-bit register is selected, only the low-order 8 bits of index register X are used and the contents of the high-order 8 bits is unchanged. In an addressing mode in which index register X is used as an index register, the address obtained by adding the contents of this register to the operand's contents is accessed.

In the **MVP**, **MVN** or **RMPA** instruction, index register X is used, also.

2.1.3 Index register Y (Y)

Index register Y is a 16-bit register with the same function as index register X. Just as in index register X, the index register length flag (x) determines whether this register is used as a 16-bit register or as an 8-bit register.

2.1.4 Stack pointer (S)

The stack pointer (S) is a 16-bit register. It is used for a subroutine call or an interrupt. It is also used when addressing modes using the stack are executed. The contents of S indicate an address (stack area) for storing registers during subroutine calls and interrupts. Bank 0_{16} is specified for the stack area. (Refer to **"2.2 Memory space.**"

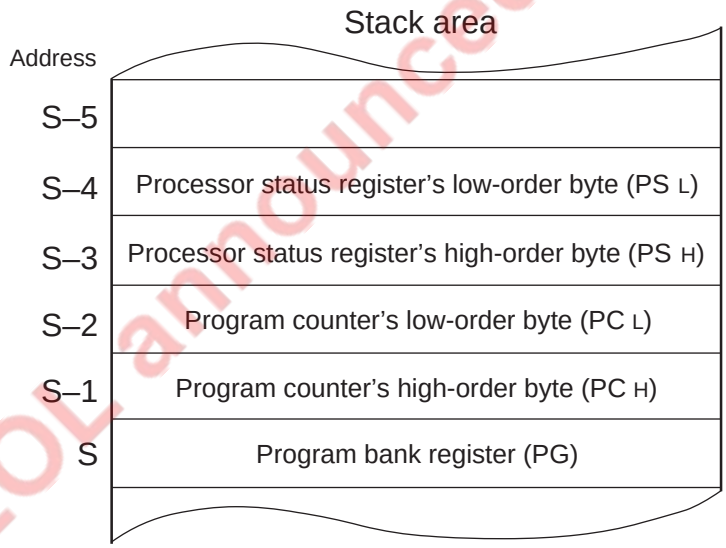
When an interrupt request is accepted, the microcomputer stores the contents of the program bank register (PG) at the address indicated by the contents of S and decrements the contents of S by 1. Then the contents of the program counter (PC) and the processor status register (PS) are stored. The contents of S after accepting an interrupt request is equal to the contents of S decremented by 5 before accepting of the interrupt request. (Refer to **Figure 2.1.2.**)

When completing the process in the interrupt routine and returning to the original routine, the contents of registers stored in the stack area are restored into the original registers in the reverse sequence (PS→PC→PG) by executing the **RTI** instruction. The contents of S is returned to the state before accepting an interrupt request.

The same operation is performed during a subroutine call, however, the contents of PS is not automatically stored. (The contents of PG may not be stored. This depends on the addressing mode.)

During interrupts or subroutine calls, the other registers are not automatically stored. Therefore, if the contents of these registers need to be held on, be sure to store them by software.

Additionally, the S's contents become "$0FFF_{16}$" at reset. The stack area changes when subroutines are nested or when multiple interrupt requests are accepted. Therefore, make sure of the subroutine's nesting depth not to destroy the necessary data.



● "S" is the initial address that the stack pointer (S) indicates at accepting an interrupt request.
The S's contents become "S – 5" after storing the above registers.

Fig. 2.1.2 Contents of stack area after accepting interrupt request

2.1.5 Program counter (PC)

The program counter is a 16-bit counter that indicates the low-order 16 bits of the address (24 bits) at which an instruction to be executed next (in other words, an instruction to be read out from an instruction queue buffer next) is stored. The contents of the high-order program counter (PC_H) become "FF_{16}," and the low-order program counter (PC_L) becomes "FE_{16}" at reset. The contents of the program counter becomes the contents of the reset's vector address (addresses $FFFE_{16}$, $FFFF_{16}$) just after reset.
Figure 2.1.3 shows the program counter and the program bank register.

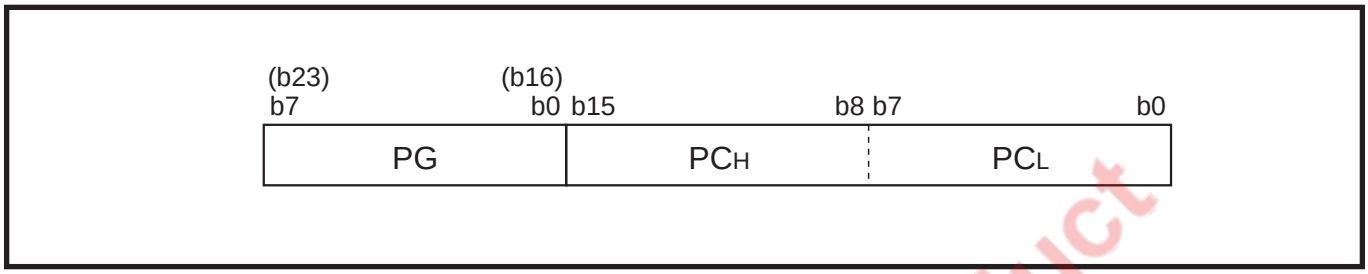


Fig. 2.1.3 Program counter and program bank register

2.1.6 Program bank register (PG)

The memory space is divided into units of 64 Kbytes. This unit is called "bank." (Refer to **"2.2 Memory space."**)
The program bank register is an 8-bit register that indicates the high-order 8 bits of the address (24 bits) at which an instruction to be executed next (in other words, an instruction to be read out from an instruction queue buffer next) is stored. These 8 bits indicate a bank.
When a carry occurs after adding the contents of the program counter or adding the offset value to the contents of the program counter in the branch instruction and others, the contents of the program bank register is automatically incremented by 1. When a borrow occurs after subtracting the contents of the program counter, the contents of the program bank register is automatically decremented by 1. Therefore, there is no need to consider bank boundaries during programming, usually.
This register is cleared to "00_{16}" at reset.

2.1.7 Data bank register (DT)

The data bank register is an 8-bit register. In the following addressing modes using the data bank register, the contents of this register is used as the high-order 8 bits (bank) of a 24-bit address to be accessed.

Use the **LDT** instruction when setting a value to this register.
This register is cleared to "00_{16}" at reset.

●Addressing modes using data bank register
- •Direct indirect
- •Direct indexed X indirect
- •Direct indirect indexed Y
- •Absolute
- •Absolute indexed X
- •Absolute indexed Y
- •Absolute bit relative
- •Stack pointer relative indirect indexed Y
- •Multiplied accumulation

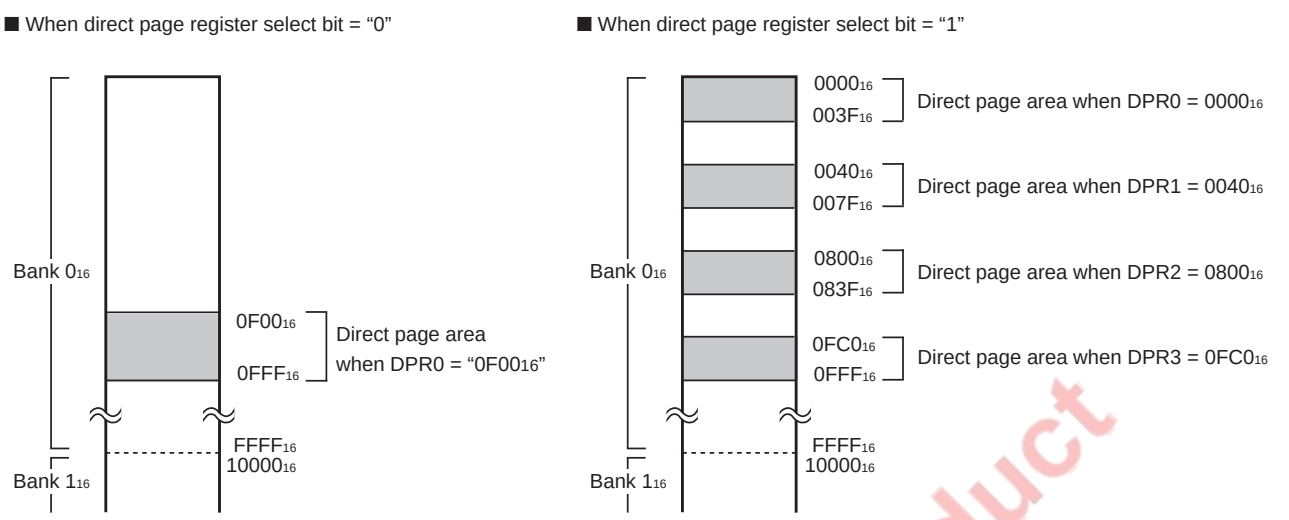
2.1.8 Direct page register 0 to 3 (DPR0 to DPR3)

The direct page register is a 16-bit register. The direct page registers (hereafter called the "DPRn") have been enhanced from the conventional 7700 Family.
These registers are used to access the 64-Kbyte space in bank 0 efficiently.
The direct page register select bit of processor mode register 1 determines whether to use DPR0 only or DPR0 through DPR3. The function of this bit is described below.

Table 2.1.1 Direct page register selection

	Direct page register select bit	
	0	1
DPRn that can be used	DPR0	DPR0 to DPR3
Block size accessible from DPRn as base address	256 bytes	64 bytes
Remarks	Compatible with conventional 7700 Family	–

Note : Once the direct page register select bit is set, do not change its value.



Notes 1: The number of cycles required to generate an address is smaller by 1 when the low-order 8 bits of DPRn are “00₁₆.”

2: If the address value obtained by adding the DPRn’s contents and offset value exceeds bank 0₁₆, the processor accesses bank 1₁₆.

Fig. 2.1.4 Direct page area selection example

When the contents of low-order 8 bits of the direct page register is “00₁₆,” the number of cycles required to generate an address is smaller by 1 than the number when its contents are not “00₁₆.” Accordingly, the access efficiency can be enhanced in this case.
This register is cleared to “0000₁₆” at reset.

●Addressing modes using direct page register
- •Direct
- •Direct indexed X
- •Direct indexed Y
- •Direct indirect
- •Direct indexed X indirect
- •Direct indirect indexed Y
- •Direct indirect long
- •Direct indirect long indexed Y
- •Direct bit relative

2.1.9 Processor status register (PS)

The processor status register is an 11-bit register.
Figure 2.1.5 shows the structure of the processor status register.

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	IPL			N	V	m	x	D	I	Z	C

Processor staus register (PS)

Note: Bits 15 to 11 are always “0” when reading. And fix each of bits 15–11 to “0” when the contents of PS is changed.

Fig. 2.1.5 Processor status register structure

(1) Bit 0: Carry flag (C)

It retains a carry or a borrow generated in the arithmetic and logic unit (ALU) during an arithmetic operation. This flag is also affected by shift and rotate instructions.
Use the **SEC** or **SEP** instruction to set this flag to “1”, and use the **CLC** or **CLP** instruction to clear it to “0”.
The contents of this flag is undefined at reset.

(2) Bit 1: Zero flag (Z)

It is set to “1” when the result of an arithmetic operation or data transfer is “0,” and cleared to “0” when otherwise. This flag is invalid in the decimal mode addition.
Use the **SEP** instruction to set this flag to “1,” and use the **CLP** instruction to clear it to “0.”
The contents of this flag is undefined at reset.

(3) Bit 2: Interrupt disable flag (I)

It disables all maskable interrupts. Interrupts are disabled when this flag is “1.” When an interrupt request is accepted, this flag is automatically set to “1” to avoid multiple interrupts. Use the **SEI** or **SEP** instruction to set this flag to “1,” and use the **CLI** or **CLP** instruction to clear it to “0.” This flag is set to “1” at reset.

(4) Bit 3: Decimal mode flag (D)

It determines whether addition and subtraction are performed in binary or decimal. Binary arithmetic is performed when this flag is “0.” When it is “1,” decimal arithmetic is performed with each 8-bit treated as 2-digit decimal (at m = 1) or each 16-bit treated as 4-digit decimal (at m = 0). Decimal adjust is automatically performed. Decimal operation is possible only with the **ADC, ADCB, SBC** and **SBCB** instructions. Use the **SEP** instruction to set this flag to “1,” and use the **CLP** instruction to clear it to “0.” This flag is cleared to “0” at reset.

(5) Bit 4: Index register length flag (x)

It determines whether each of index register X and index register Y is used as a 16-bit register or an 8-bit register. That register is used as a 16-bit register when this flag is “0,” and as an 8-bit register when it is “1” **(Note)**. Use the **SEP** instruction to set this flag to “1,” and use the **CLP** instruction to clear it to “0.” This flag is cleared to “0” at reset.

Note: When transferring data between registers which are different in bit length, the data is transferred with the length of the destination register, but except for the **TXA**, **TYA**, **TXB**, **TYB**, and **TXS** instructions.

(6) Bit 5: Data length flag (m)

It determines whether to use data as a 16-bit unit or as an 8-bit unit. A data is treated as a 16-bit unit when this flag is "0," and as an 8-bit unit when it is "1" **(Note)**.
Use the **SEM** or **SEP** instruction to set this flag to "1," and use the **CLM** or **CLP** instruction to clear it to "0." This flag is cleared to "0" at reset.

Note: When transferring data between registers which are different in bit length, the data is transferred with the length of the destination register, but except for the **TXA**, **TYA**, **TXB**, **TYB**, and **TXS** instructions.

(7) Bit 6: Overflow flag (V)

It is used when adding or subtracting with a word regarded as signed binary. The overflow flag is set to "1" when the result of addition or subtraction exceeds the range between –2147483648 and +2147483647 (when 32-bit length operation), the range between –32768 and +32767 (when 16-bit length operation), or the range between –128 and +127 (when 8-bit length operation).
The overflow flag is also set to "1" when the result of division exceeds the length of the register which will store the result, in the **DIV** or **DIVS** instruction. This flag is invalid in the decimal mode. Use the **SEP** instruction to set this flag to "1," and use the **CLV** or **CLP** instruction to clear it to "0."
The contents of this flag is undefined at reset.

(8) Bit 7: Negative flag (N)

It is set to "1" when the result of arithmetic operation or data transfer is negative. (The most significant bit of the result is "1.") It is cleared to "0" in all other cases. This flag is invalid in the decimal mode.
Use the **SEP** instruction to set this flag to "1," and use the **CLP** instruction to clear it to "0."
The contents of this flag is undefined at reset.

(9) Bits 10 to 8: Processor interrupt priority level (IPL)

These 3 bits can determine the processor interrupt priority level to one of levels 0 to 7. The interrupt is enabled when the interrupt priority level of a required interrupt, which is set in each interrupt control register, is higher than IPL. When an interrupt request is accepted, IPL is stored in the stack area, and IPL is replaced by the interrupt priority level of the accepted interrupt request.
There are no instruction to directly set or clear the bits of IPL. IPL can be changed by storing the new IPL into the stack area and updating the processor status register with the **PUL** or **PLP** instruction.
The contents of IPL is cleared to "000_2" at reset.

2.2 Access space

The memory space of the 7900 Series is a 16-Mbyte space from addresses 0_{16} to $FFFFFF_{16}$. (Refer to the **Figure 2.2.1**.) However, addresses $FF0000_{16}$ to $FFFFFF_{16}$ cannot be used because this area is reserved. A 24-bit address is generated by combination of the program counter (PC), which is 16 bits of structure, and the program bank register (PG), which is 8 bits of structure. The memory space of the 7900 Series is divided into units of 64 Kbytes. This unit is called “bank.” The PG indicates the bank number.
The memory and I/O devices are assigned in the same access space. Accordingly, it is possible to perform transfer and arithmetic operations using the same instructions without discrimination of the memory from I/O devices.

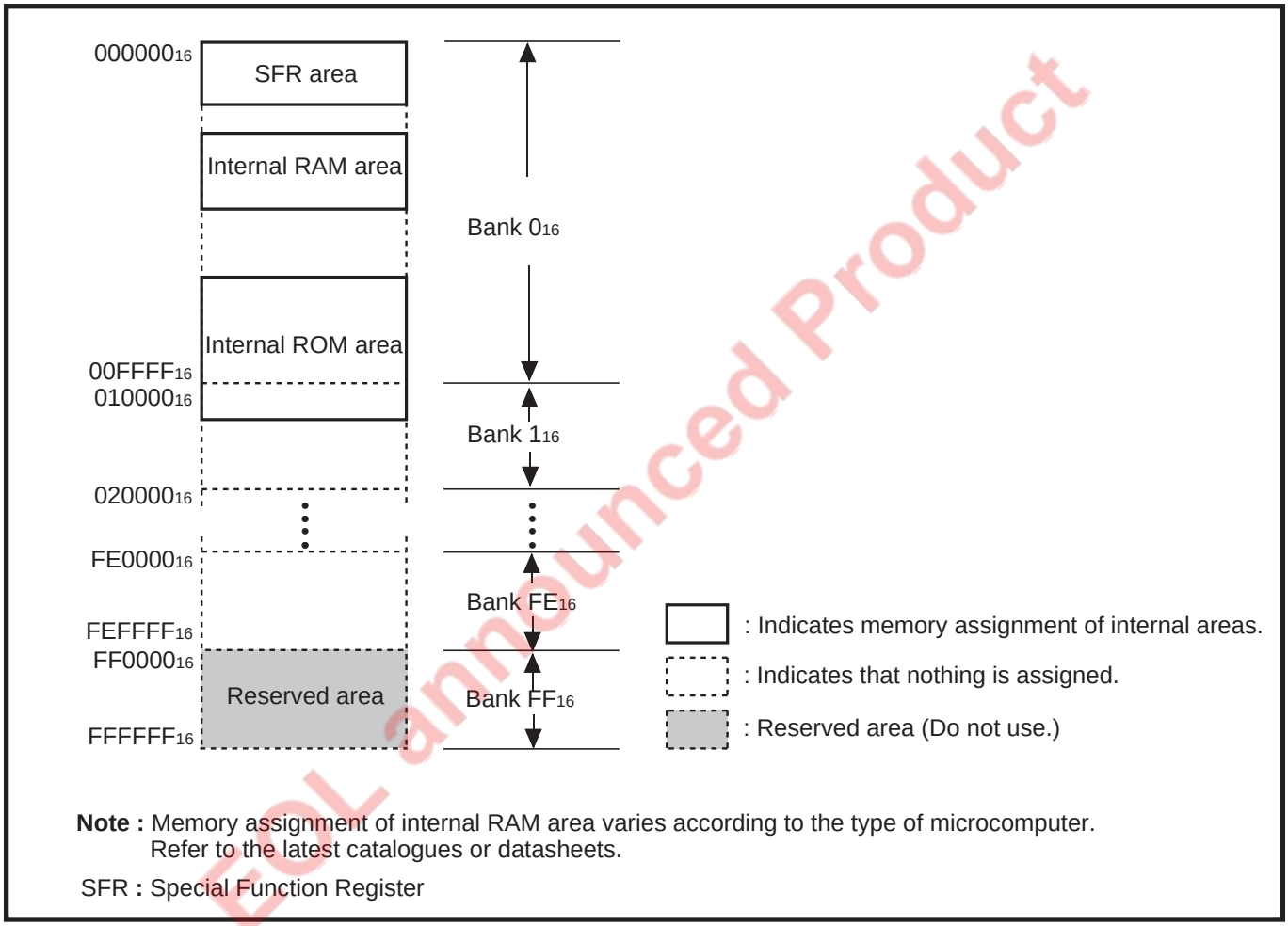


Fig. 2.2.1 7900 Series’s access space

2.3 Addressing modes

2.3.1 Overview

To execute an instruction, when the data required for the operation is retrieved from a memory or the result of the operation is stored to it, it is necessary to specify the address of the memory location in advance. Address specification is also necessary when the control is to jump to a certain memory address during program execution. Addressing means the method of specifying the memory address.
The memory access of the 7900 Series microcomputers is reinforced with 27 different addressing modes.

2.3.2 Explanation of addressing modes

Each addressing mode is explained on the corresponding page indicated below:

Note: Unless otherwise noted, in each explanation diagram for the addressing mode of which name includes “direct,” “Direct page register” means DPR0 only.

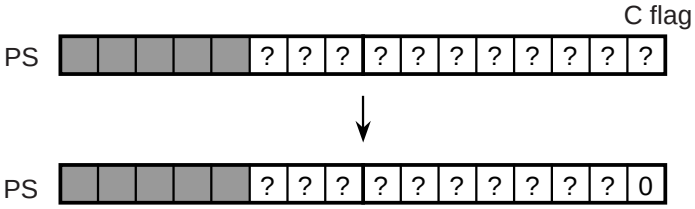
Implied

Mode : Implied addressing mode

Function : These instructions do not have an operand in the mnemonic.

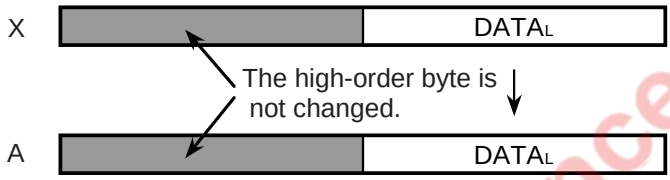
ex. : Mnemonic
CLC

Machine code
14_{16}



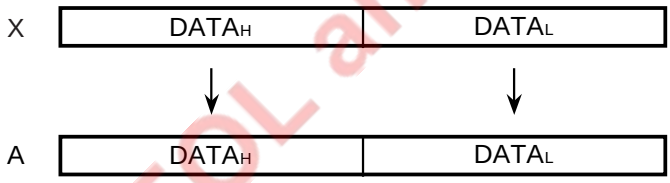
ex. : Mnemonic
TXA
(m="1," x="1")

Machine code
$A4_{16}$



ex. : Mnemonic
TXA
(m="0," x="0")

Machine code
$A4_{16}$



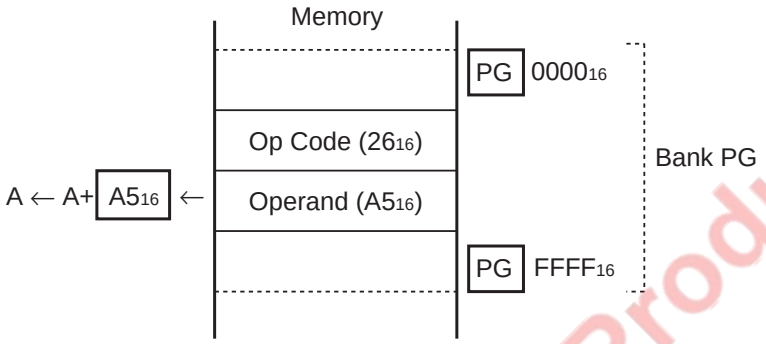
Immediate

Mode : Immediate addressing mode

Function : These instructions operate with a register and a immediate value.

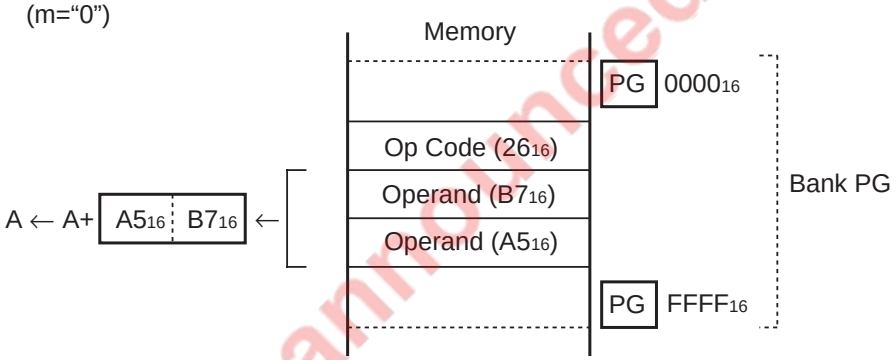
ex. : Mnemonic
ADD A, #0A5H
(m="1")

Machine code
26_{16} $A5_{16}$



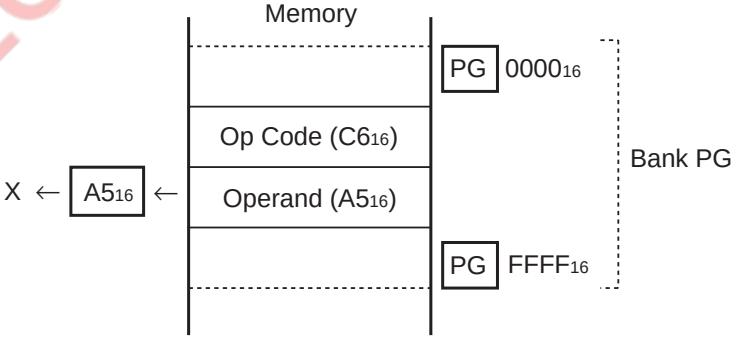
ex. : Mnemonic
ADD A, #0A5B7H
(m="0")

Machine code
26_{16} $B7_{16}$ $A5_{16}$

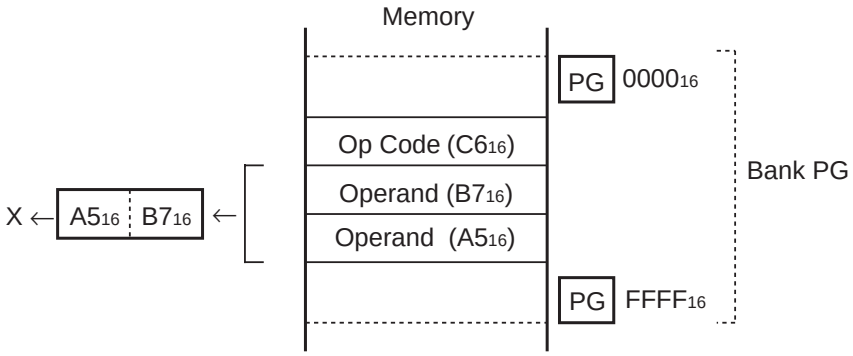


ex. : Mnemonic
LDX #0A5H
(x="1")

Machine code
$C6_{16}$ $A5_{16}$



ex. : Mnemonic Machine code
LDX #0A5B7H $C6_{16}$ $B7_{16}$ $A5_{16}$
(x="0")

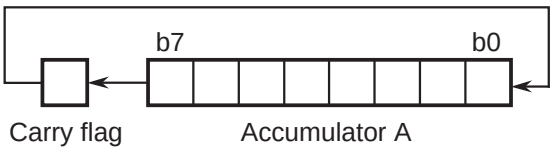


Mode : Accumulator addressing mode

Function : These instructions manipulate the contents of an accumulator.

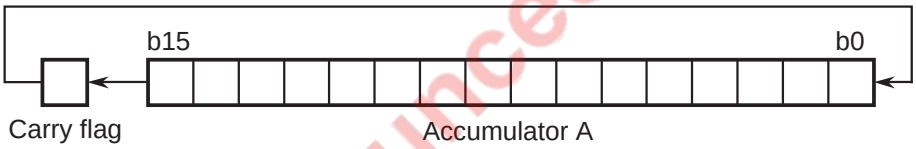
ex. : Mnemonic — Machine code

ROL A (m="1") — 13_{16}



ex. : Mnemonic — Machine code

ROL A (m="0") — 13_{16}



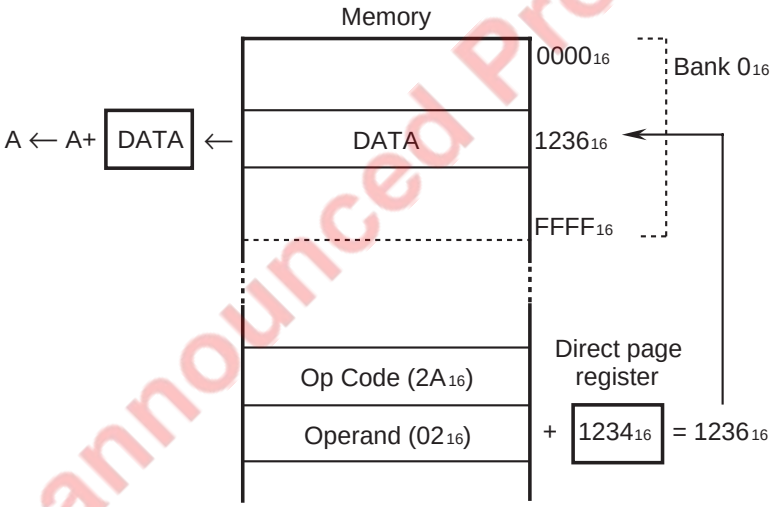
Mode : Direct addressing mode

Function : The memory contents in bank 0 specified by the result of adding the instruction's operand and the contents of the direct page register are an actual data. However, if the value derived by adding the instruction's operand and the direct page register's content's exceeds the bank 0_{16} range, memory in bank 1 is specified.
The direct page register select bit of processor mode register 1 allows the user to choose one of the following options :
- Use direct page register 0 (DPR0) only.
  In this case, specify the offset from DPR0 in length of 8 bits.
- Use direct page registers 0 through 3 (DPR0 through 3).
  In this case, use the high-order 2 bits of the operand (8 bits) to specify the direct page register and the low-order 6 bits to specify the offset.

< Diect addressing mode>

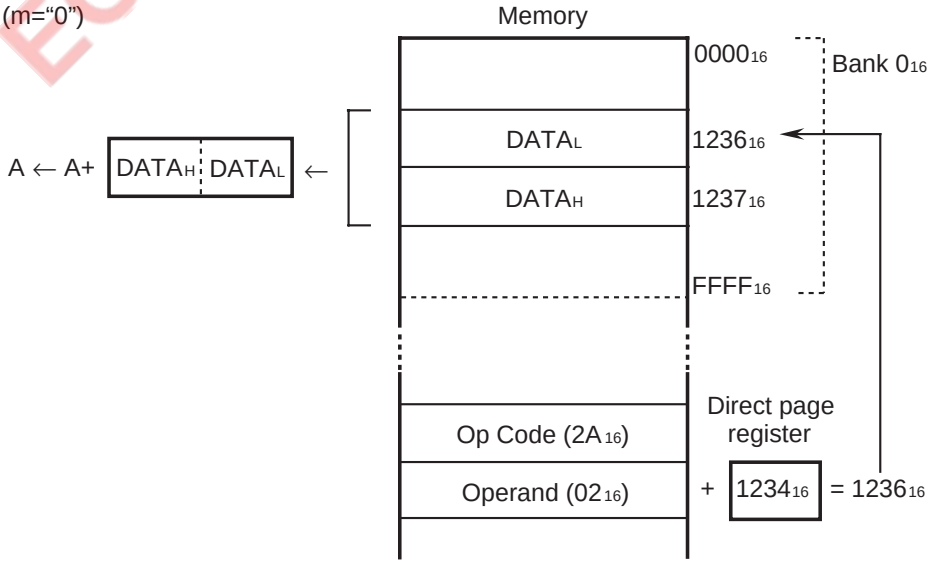
ex. : Mnemonic
ADD A, 02H
(m="1")

Machine code
$2A_{16}$ 02_{16}



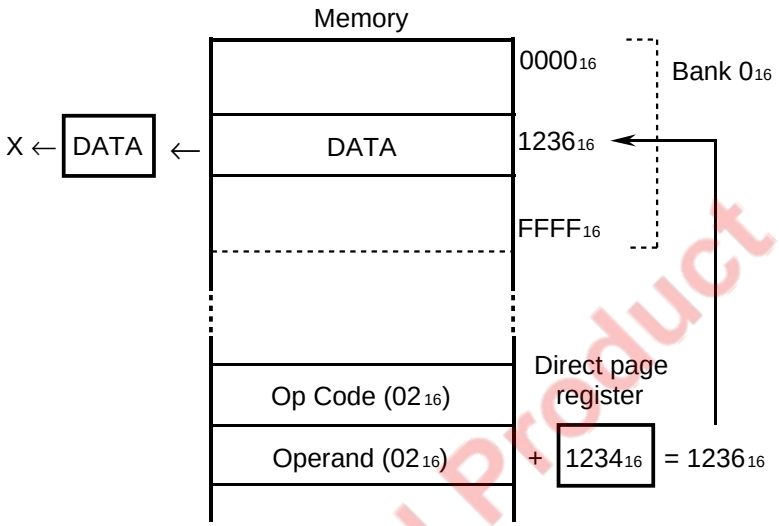
ex. : Mnemonic
ADD A, 02H
(m="0")

Machine code
$2A_{16}$ 02_{16}



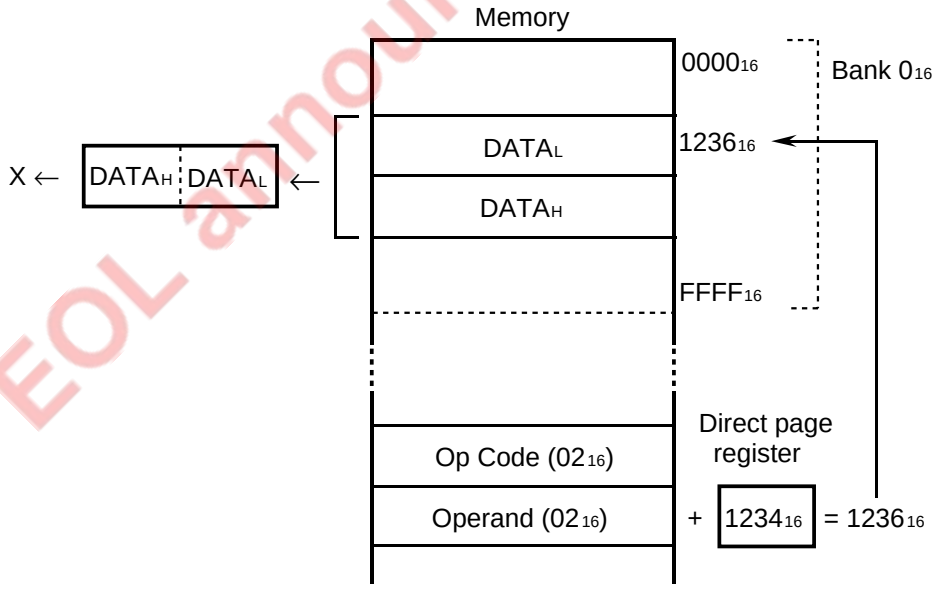
ex. : Mnemonic
LDX 02H
(x="1")

Machine code
02_{16} 02_{16}



ex. : Mnemonic
LDX 02H
(x="0")

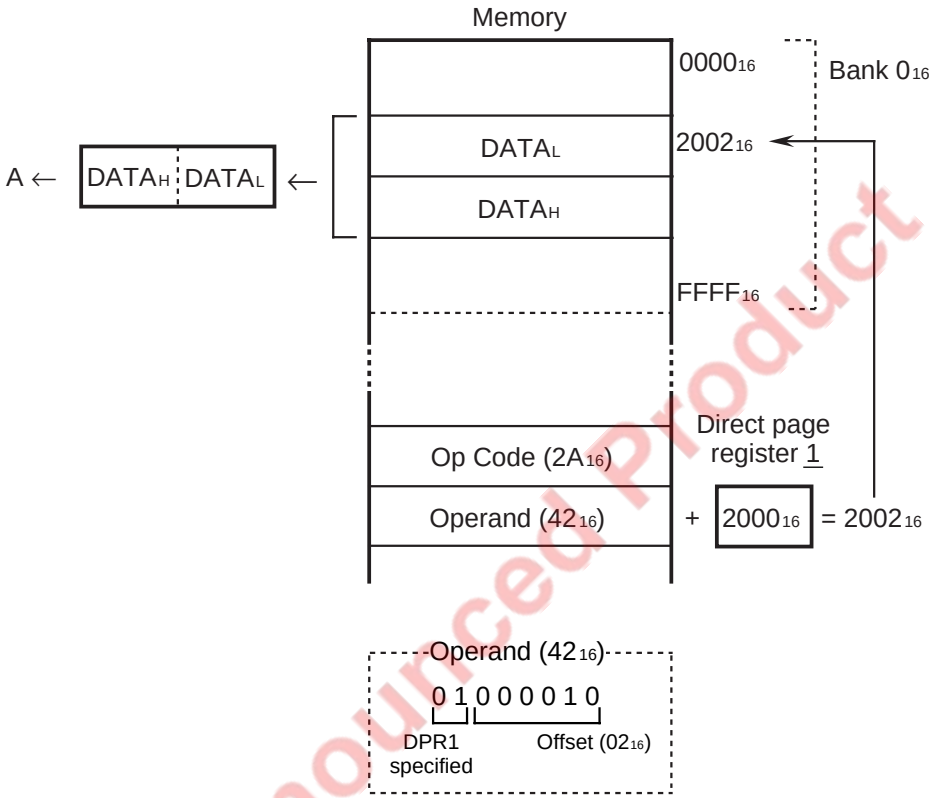
Machine code
02_{16} 02_{16}



<Extension direct addressing mode>

ex. : Mnemonic
ADD A, 42H
(x="0")

Machine code
$2A_{16}$ 42_{16}



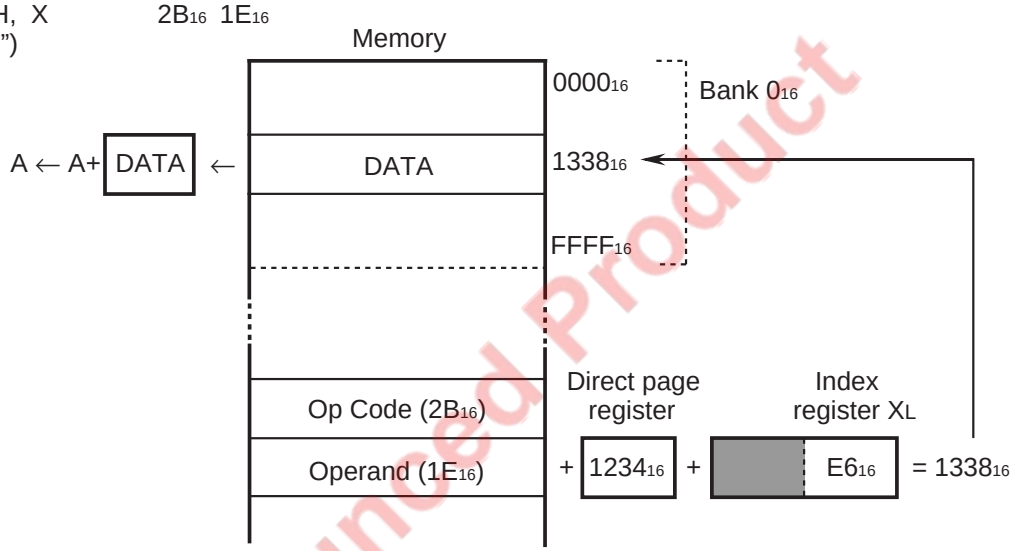
Direct Indexed X

Mode : Direct indexed X addressing mode

Function : The contents of a memory in bank 0_{16} are an actual data. This memory location is specified by the result of adding the instruction's operand, the direct page register's contents and the index register X's contents. When, however, the result of adding the instruction's operand, the direct page register's contents and the index register X's contents exceeds the bank 0_{16} or bank 1_{16} range, the memory location in bank 1_{16} or bank 2_{16} is specified.

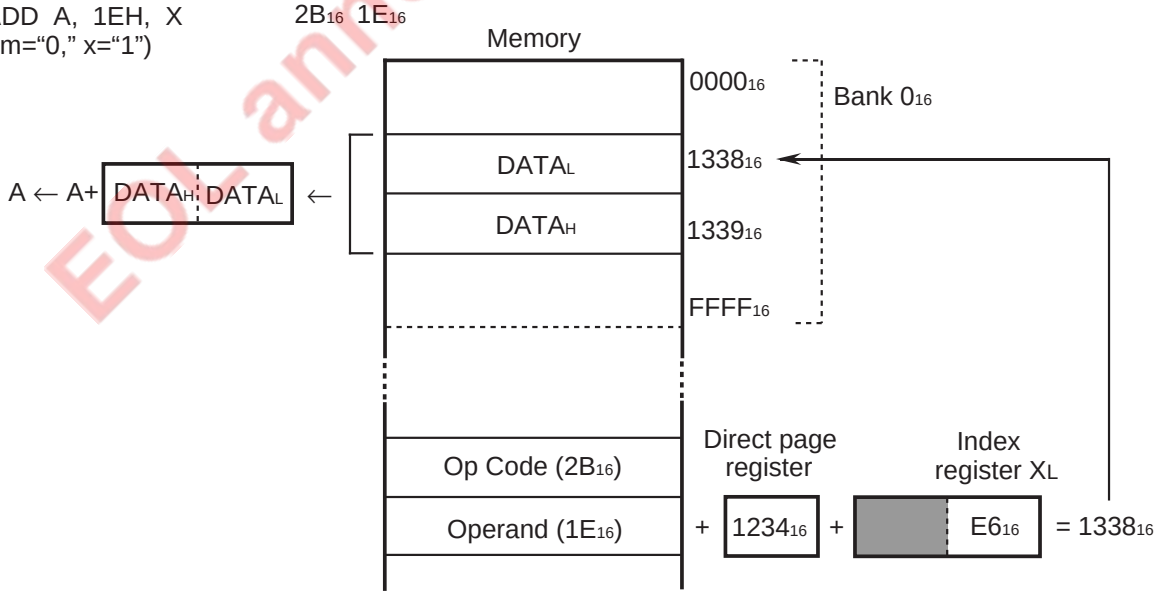
ex. : Mnemonic
ADD A, 1EH, X
(m="1," x="1")

Machine code
$2B_{16}$ $1E_{16}$



ex. : Mnemonic
ADD A, 1EH, X
(m="0," x="1")

Machine code
$2B_{16}$ $1E_{16}$



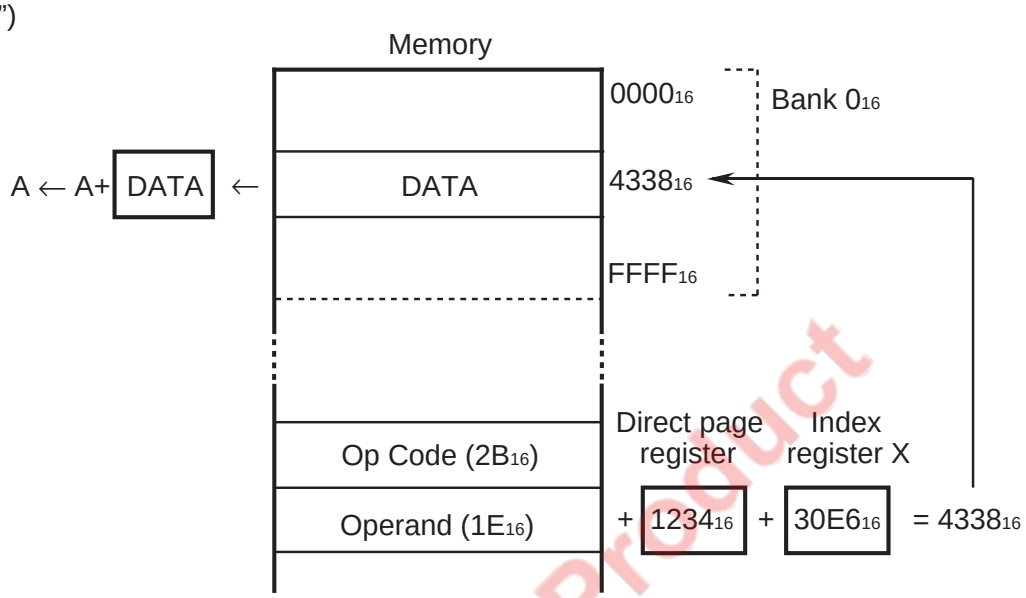
Direct Indexed X

ex. : Mnemonic

ADD A, 1EH, X
(m="1," x="0")

Machine code

$2B_{16}$ $1E_{16}$

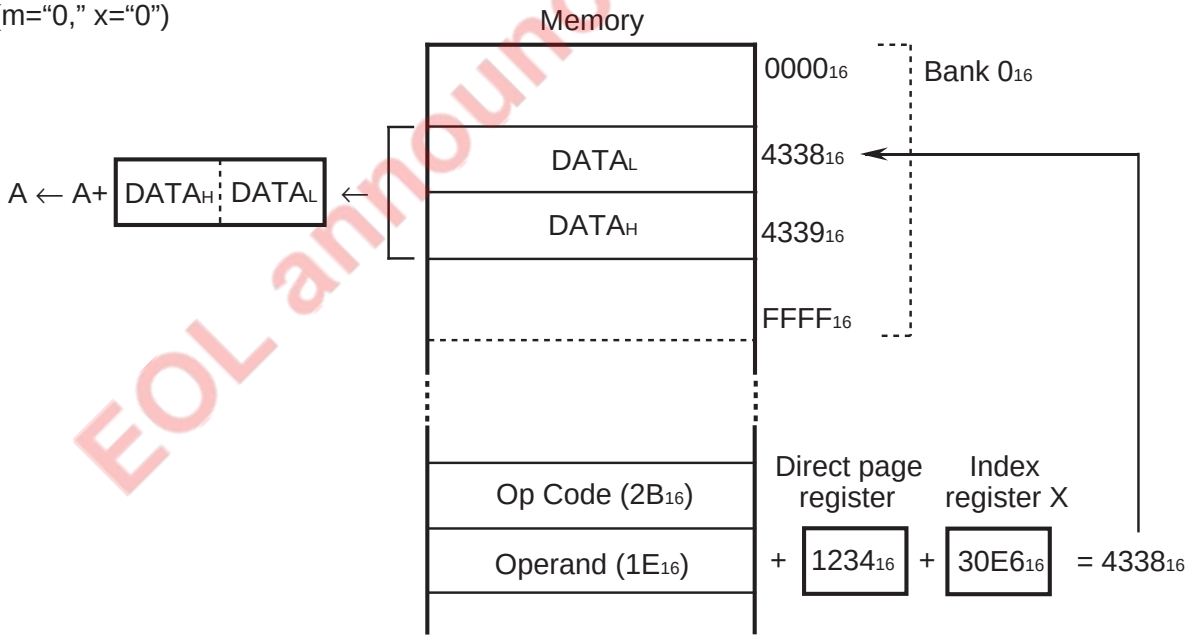


ex. : Mnemonic

ADD A, 1EH, X
(m="0," x="0")

Machine code

$2B_{16}$ $1E_{16}$

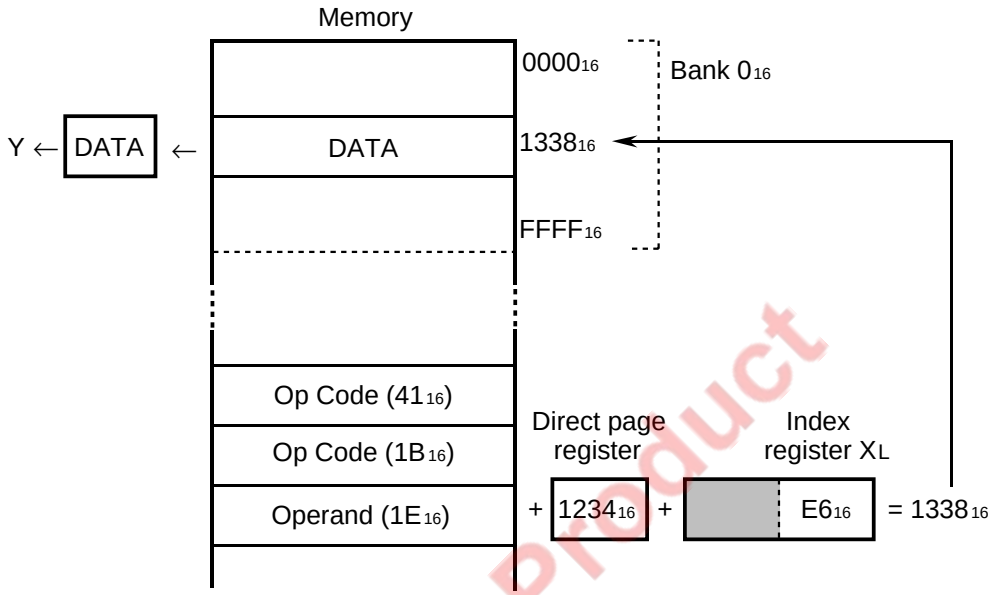


ex. : Mnemonic

LDY 1EH, X
(x="1")

Machine code

41_{16} $1B_{16}$ $1E_{16}$

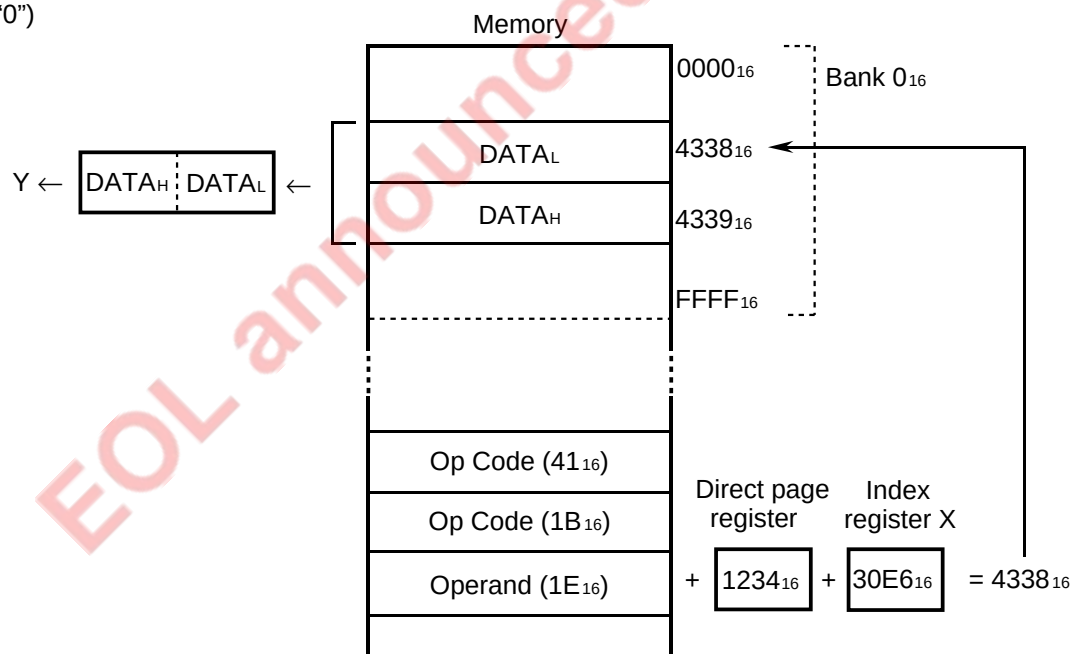


ex. : Mnemonic

LDY 1EH, X
(x="0")

Machine code

41_{16} $1B_{16}$ $1E_{16}$



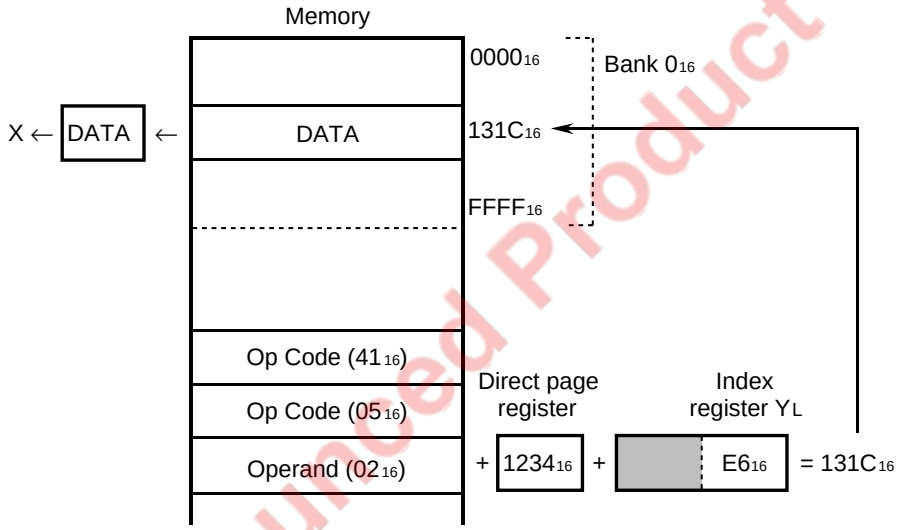
Direct Indexed Y

Mode : Direct indexed Y addressing mode

Function : The contents of a memory in bank 0_{16} are an actual data. This memory location is specified by the result of adding the instruction's operand, the direct page register's contents and the index register Y's contents. When, however, the result of adding the instruction's operand, the direct page register's contents and the index register Y's contents exceeds the bank 0_{16} or bank 1_{16} range, the memory location in bank 1_{16} or bank 2_{16} is specified.

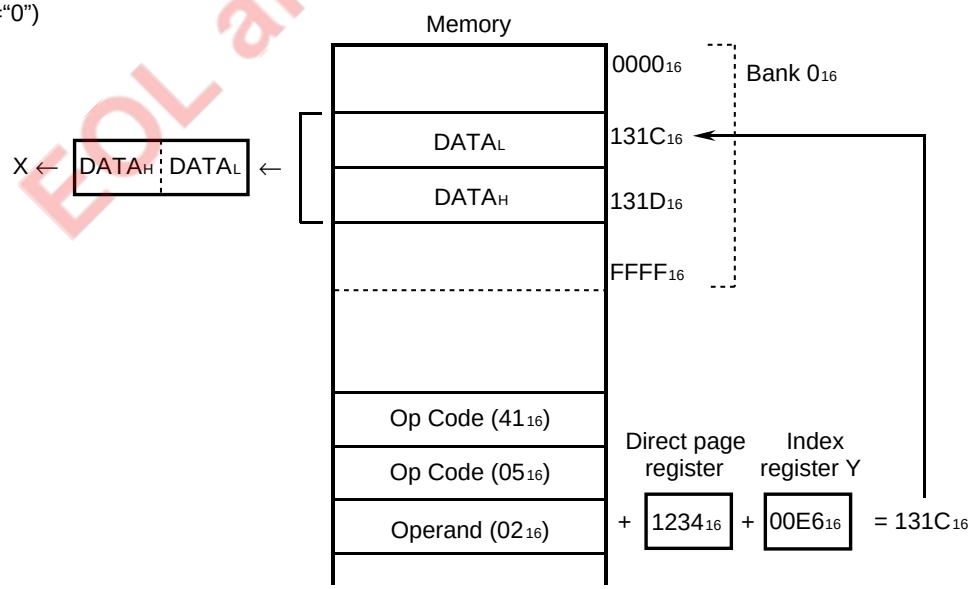
ex. : Mnemonic
LDX 02H, Y
(x="1")

Machine code
41_{16} 05_{16} 02_{16}



ex. : Mnemonic
LDX 02H, Y
(x="0")

Machine code
41_{16} 05_{16} 02_{16}



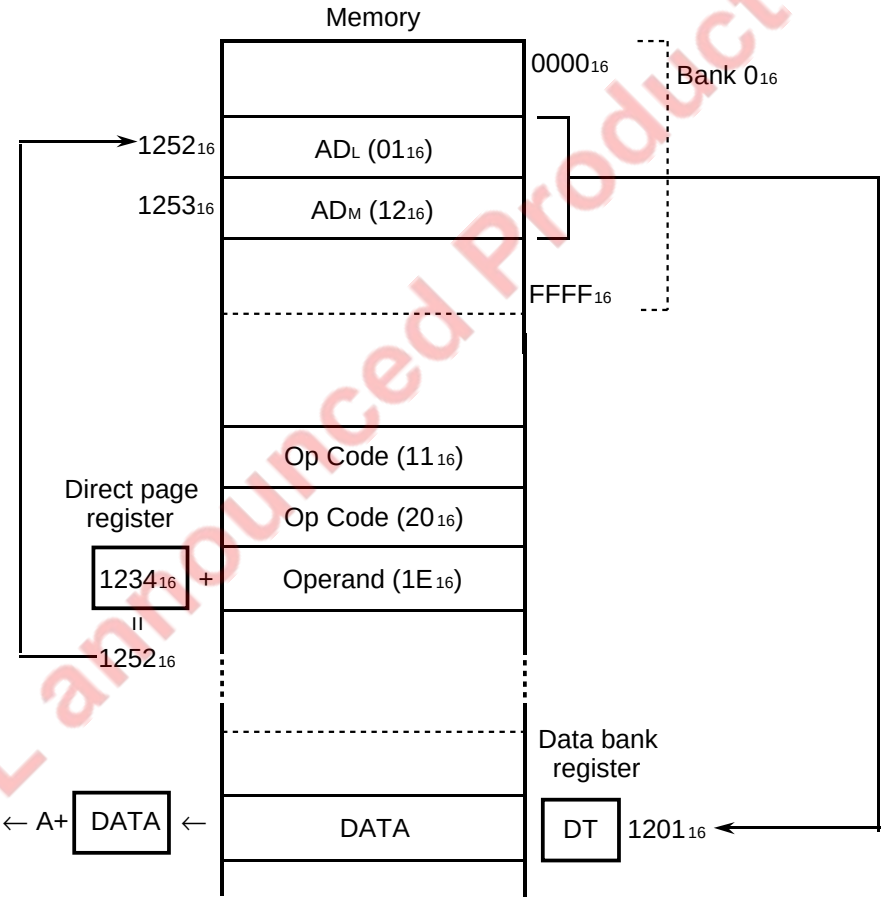
Direct Indirect

Mode : Direct indirect addressing mode

Function : Specifies a sequence of 2-byte memory in bank 0_{16} by the result of adding the instruction's operand to the direct page register's contents. The contents of the memory location specified by these 2 bytes in bank DT (DT is the data bank register's contents) are an actual data. When, however, the result of adding the instruction's operand to the direct page register's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified.

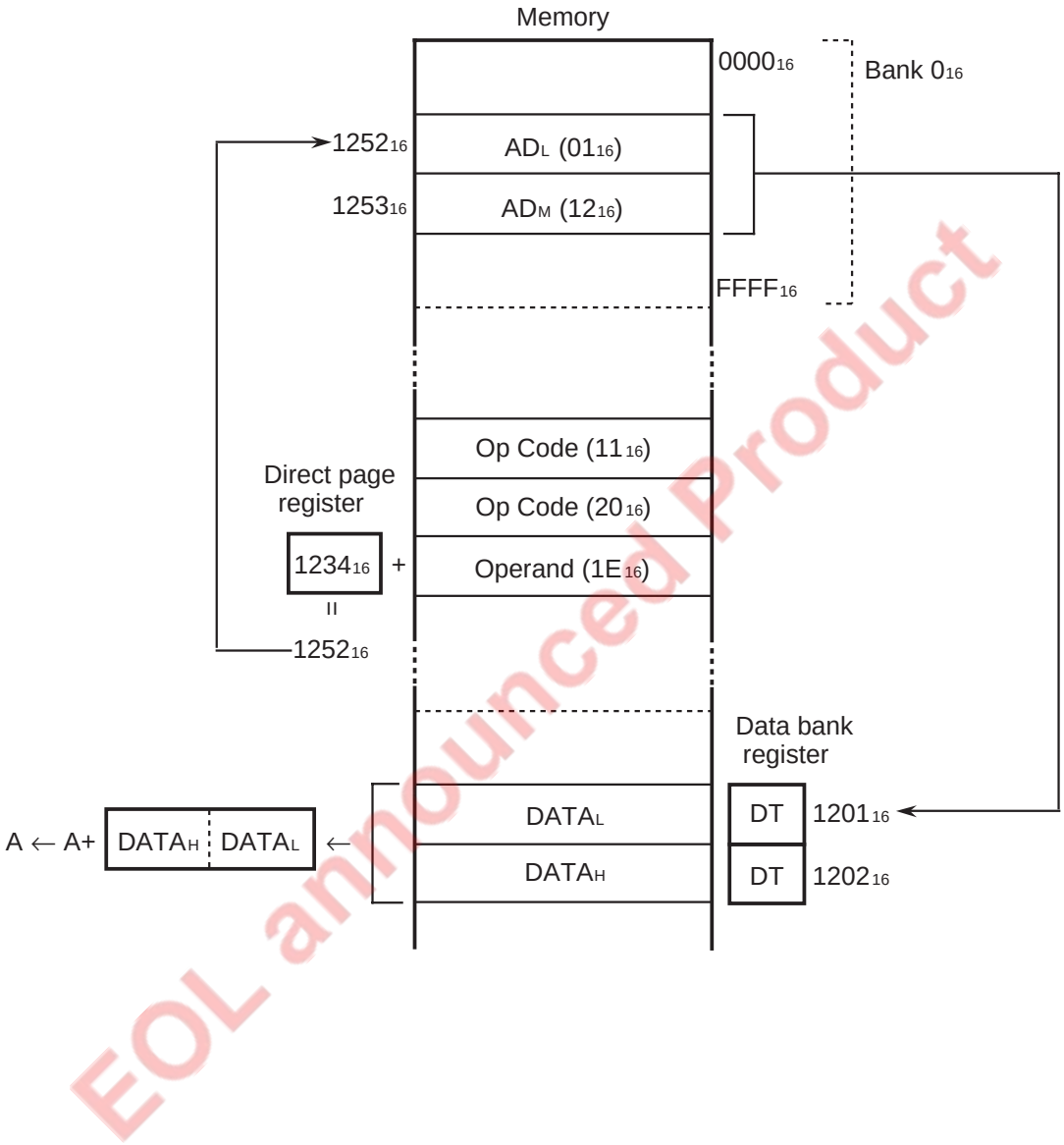
ex. : Mnemonic
ADD A, (1EH)
(m="1")

Machine code
11_{16} 20_{16} $1E_{16}$



ex. : Mnemonic
ADD A, (1EH)
(m="0")

Machine code
11_{16} 20_{16} $1E_{16}$



Direct Indexed X Indirect

Mode : Direct indexed X indirect addressing mode

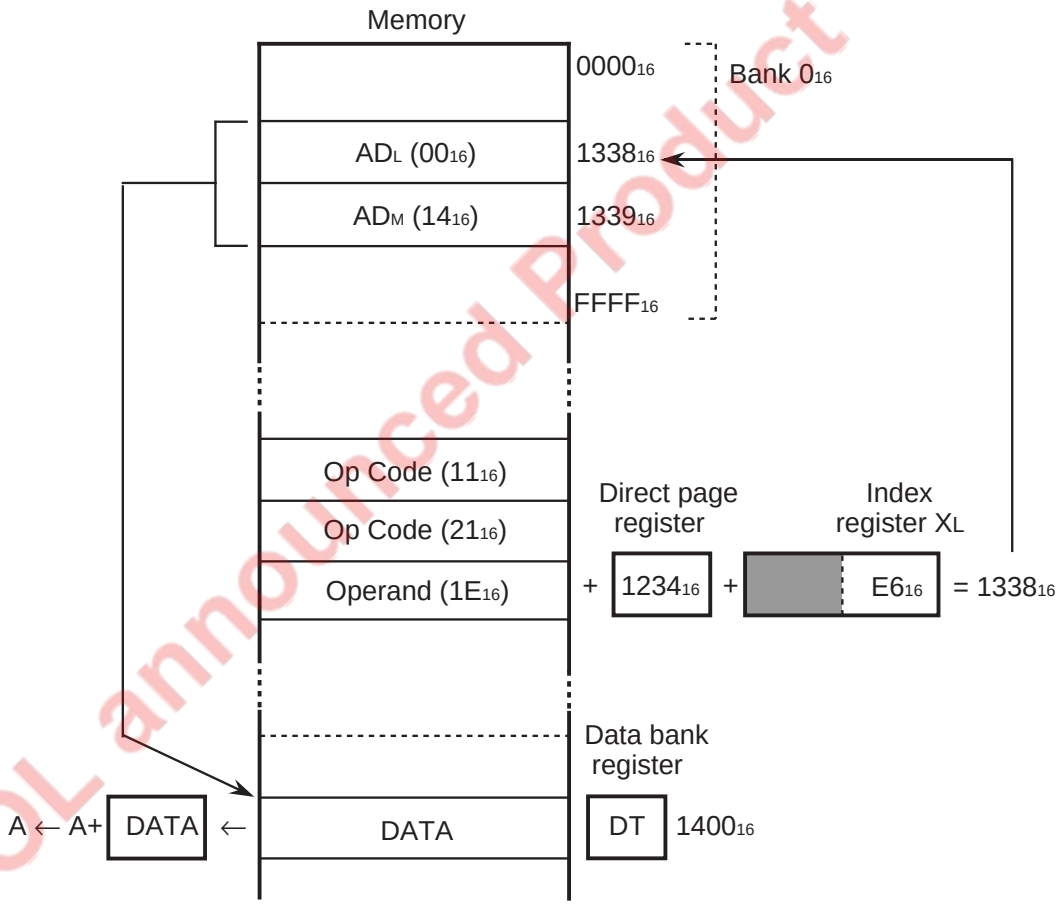
Function : Specifies a sequence of 2-byte memory in bank 0_{16} by the result of adding the instruction's operand, the direct page register's contents and the index register X's contents. The contents of the memory location specified by these bytes in bank DT (DT is the data bank register's contents) are an actual data. When, however, the result of adding the instruction's operand, the direct page register's contents and the index register X's contents exceeds the bank 0_{16} or bank 1_{16} range, the memory location in bank 1_{16} or bank 2_{16} is specified.

ex. : Mnemonic

ADD A, (1EH, X)
(m="1," x="1")

Machine code

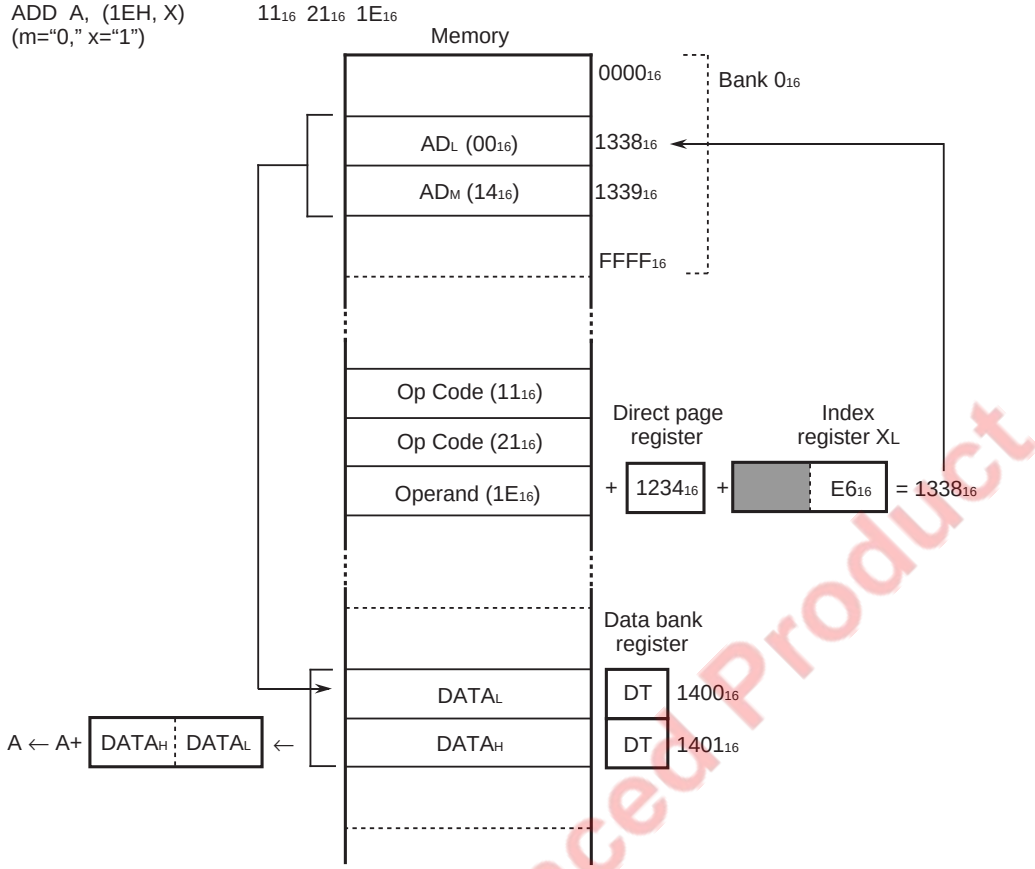
11_{16} 21_{16} $1E_{16}$



Direct Indexed X Indirect

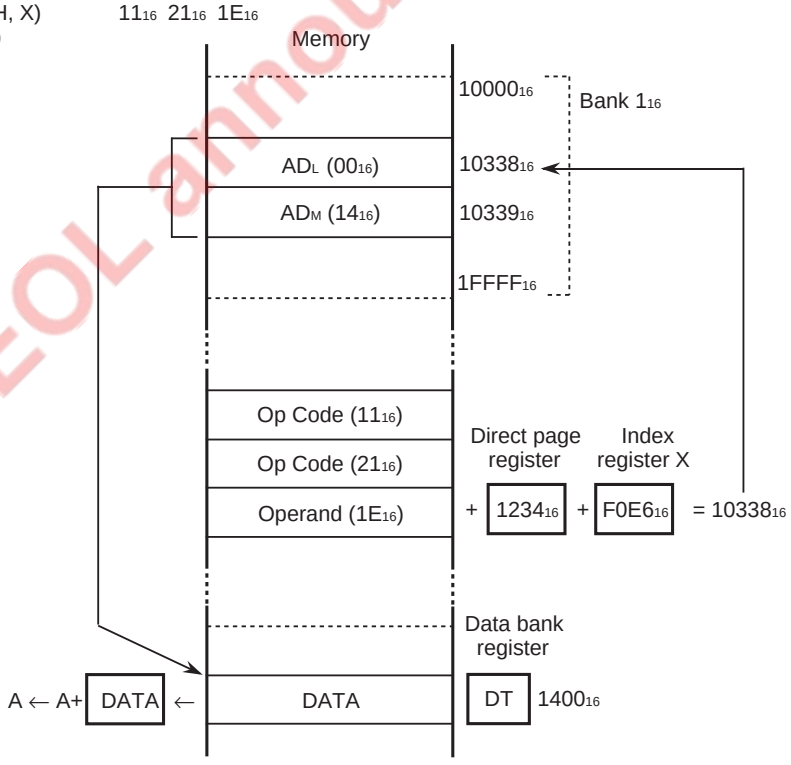
ex. : Mnemonic
ADD A, (1EH, X)
(m="0," x="1")

Machine code
11_{16} 21_{16} $1E_{16}$



ex. : Mnemonic
ADD A, (1EH, X)
(m="1," x="0")

Machine code
11_{16} 21_{16} $1E_{16}$



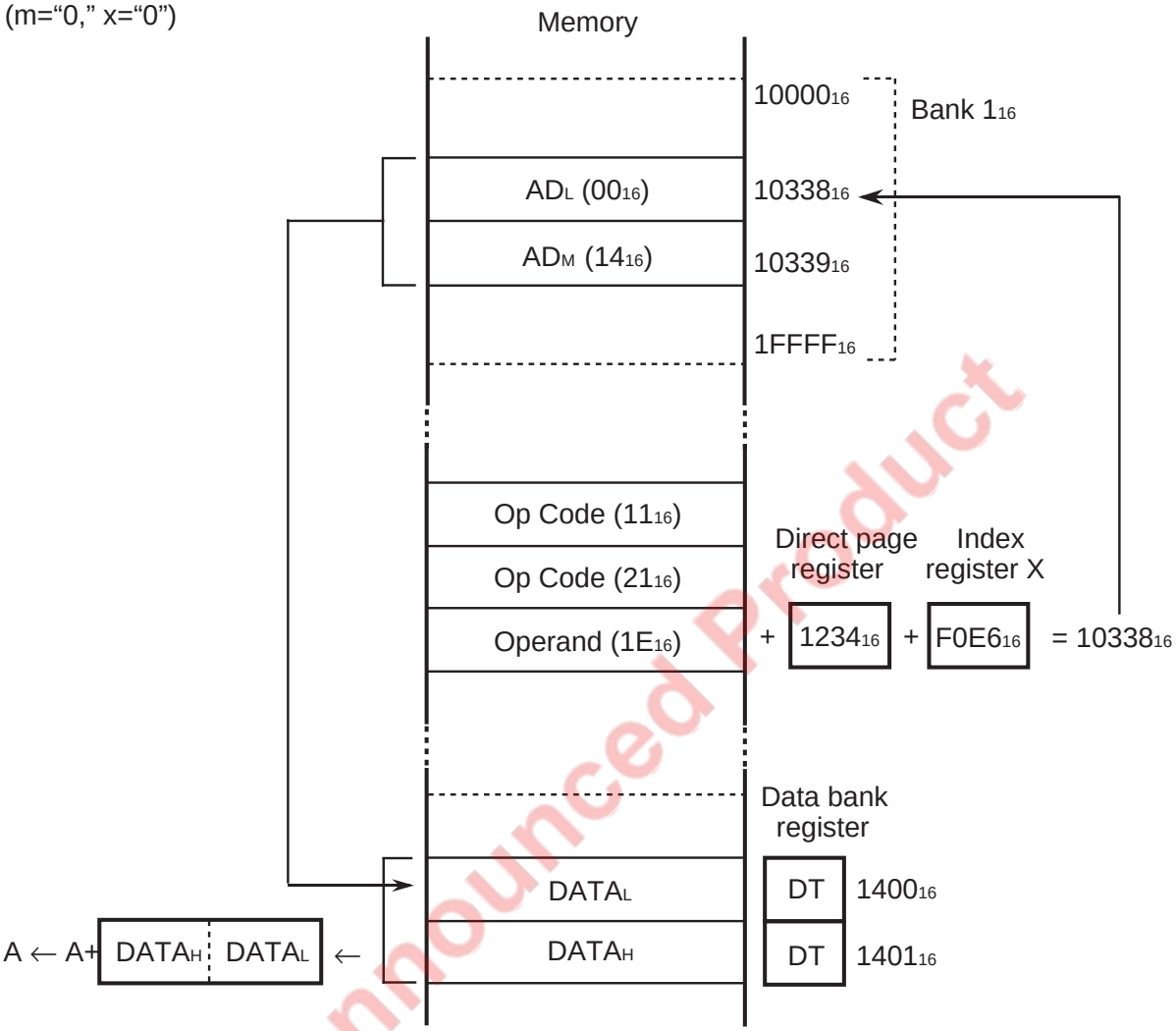
Direct Indexed X Indirect

ex. : Mnemonic

ADD A, (1EH, X)
(m="0," x="0")

Machine code

11_{16} 21_{16} $1E_{16}$



Direct Indirect Indexed Y

Mode : Direct indirect indexed Y addressing mode

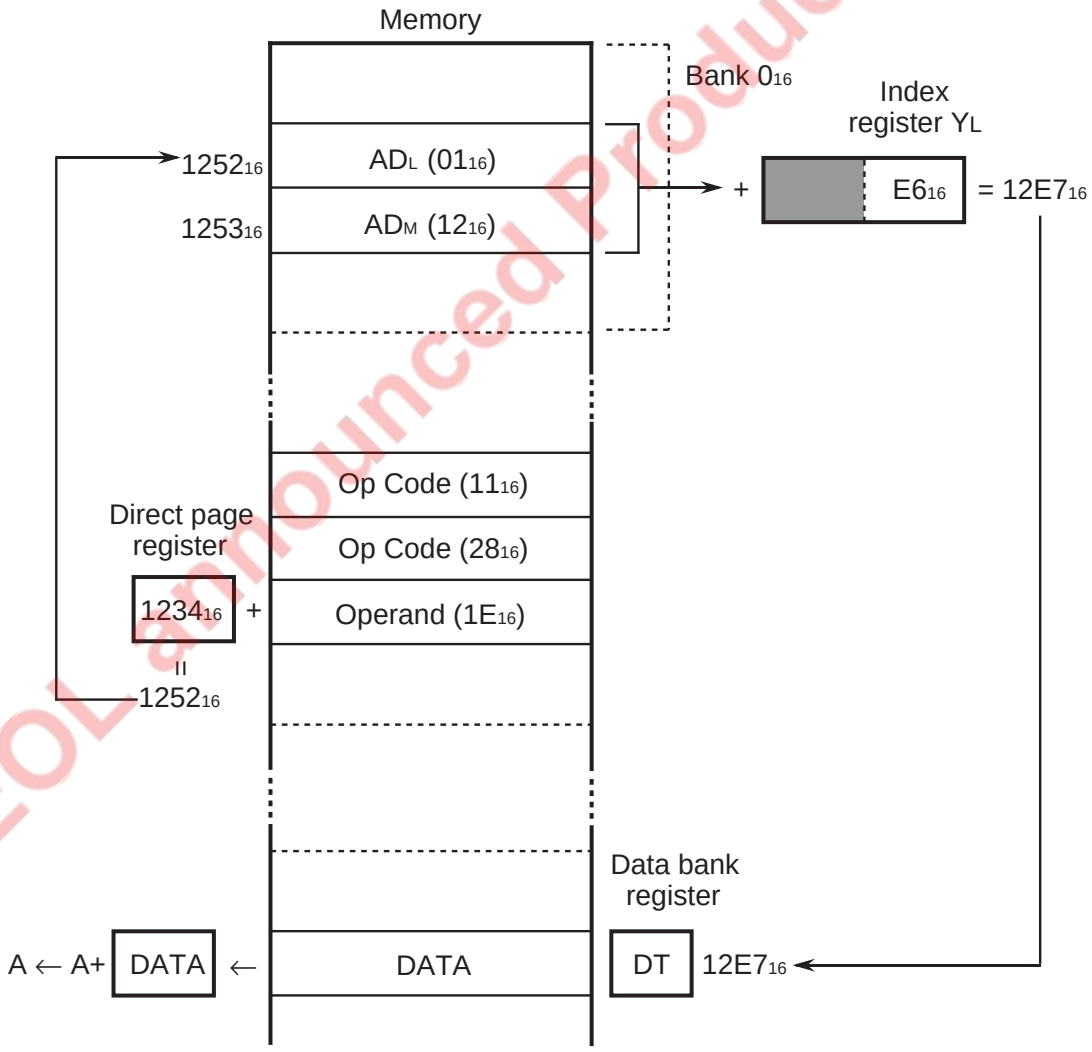
Function : Specifies a sequence of 2-byte memory in bank 0_{16} by the result of adding the instruction's operand to the direct page register's contents. The following is an actual data: the contents of the memory location specified by the result of adding the contents of these 2 bytes to the index register Y's contents and the contents of the data bank register. When, however, the result of adding the instruction's operand to the direct page register's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified. Additionally, if the addition of the memory's contents and the index register Y's contents generates a carry, the value which is 1 larger than the contents of the data bank register indicates the bank.

ex. : Mnemonic

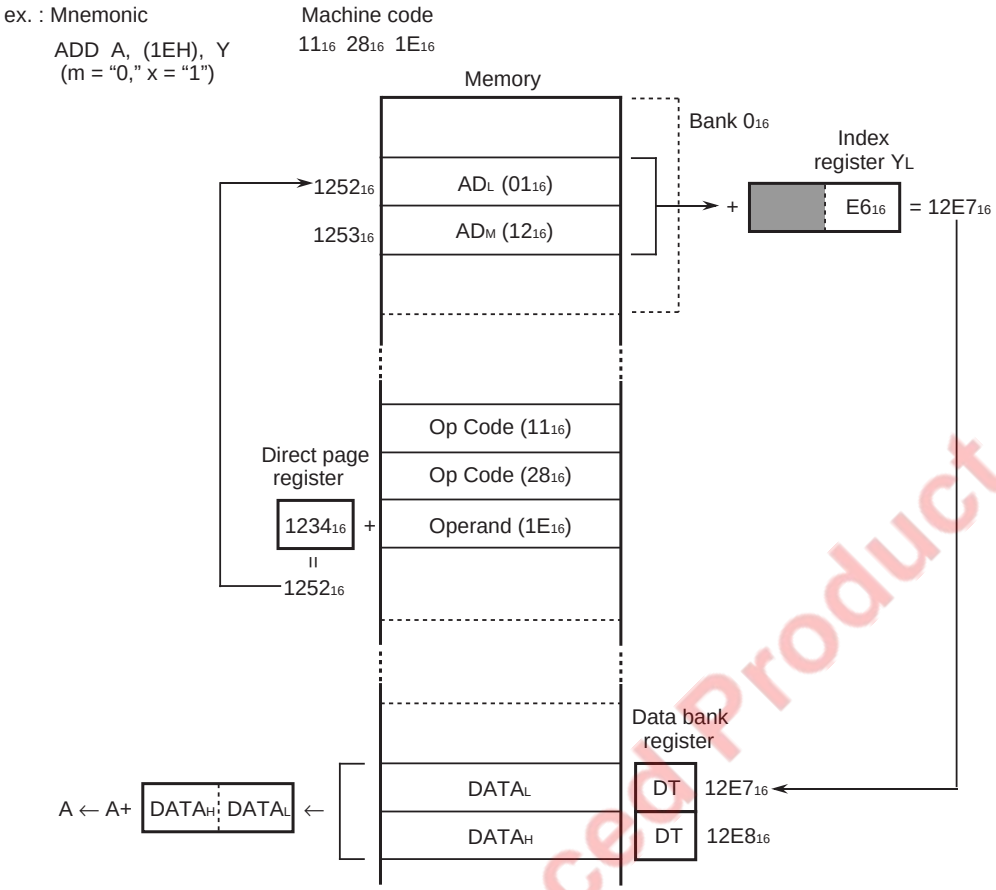
ADD A, (1EH), Y
(m = "1," x = "1")

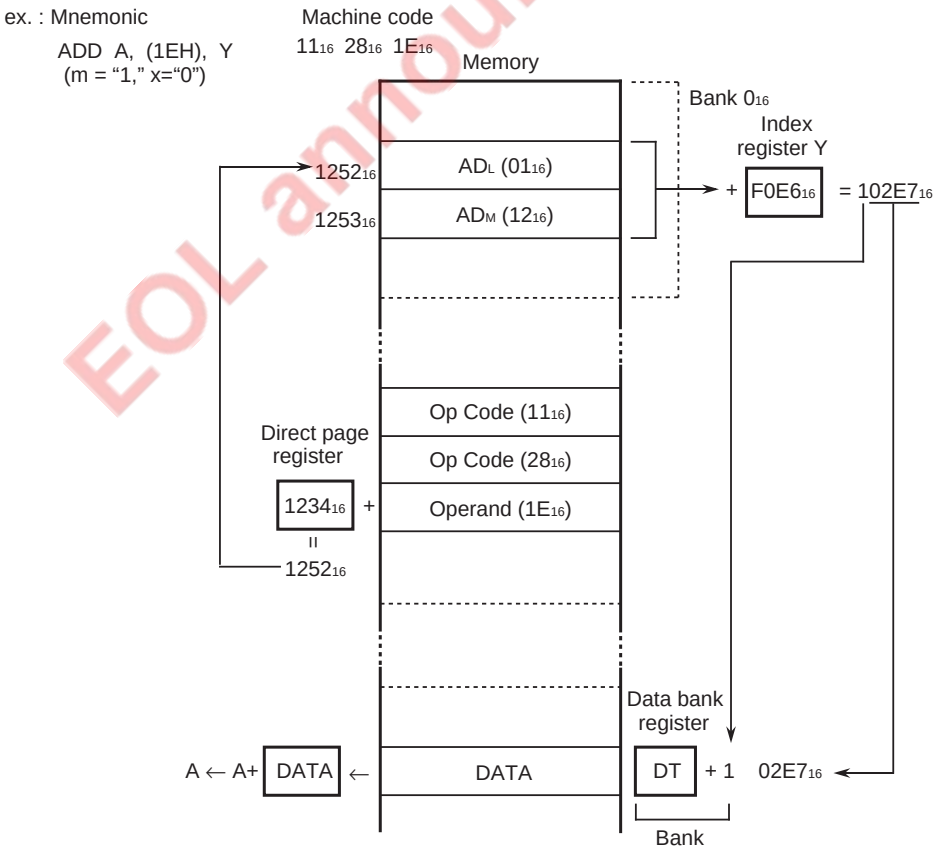
Machine code

11_{16} 28_{16} $1E_{16}$



Direct Indirect Indexed Y





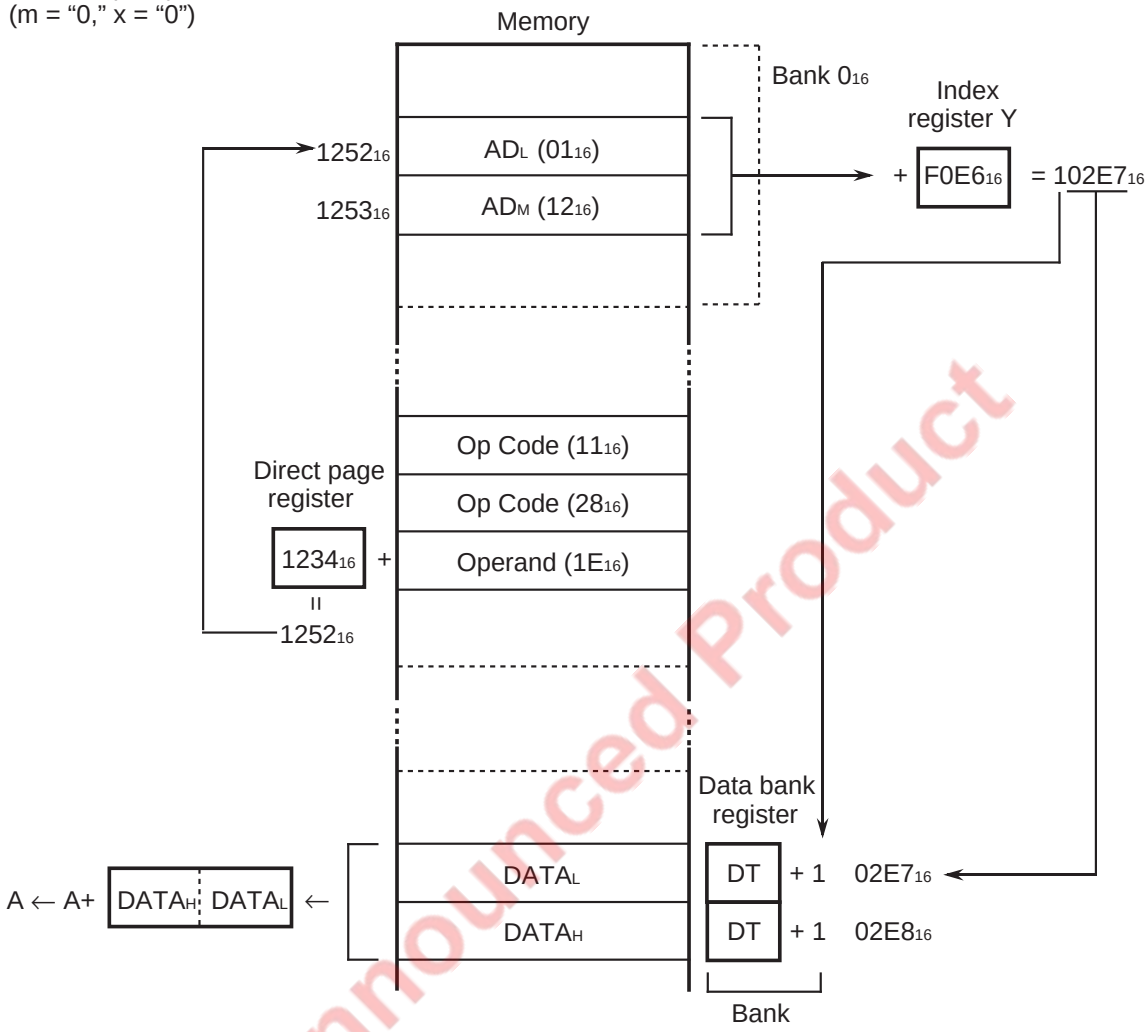
Direct Indirect Indexed Y

ex. : Mnemonic

ADD A, (1EH), Y
(m = "0," x = "0")

Machine code

11_{16} 28_{16} $1E_{16}$



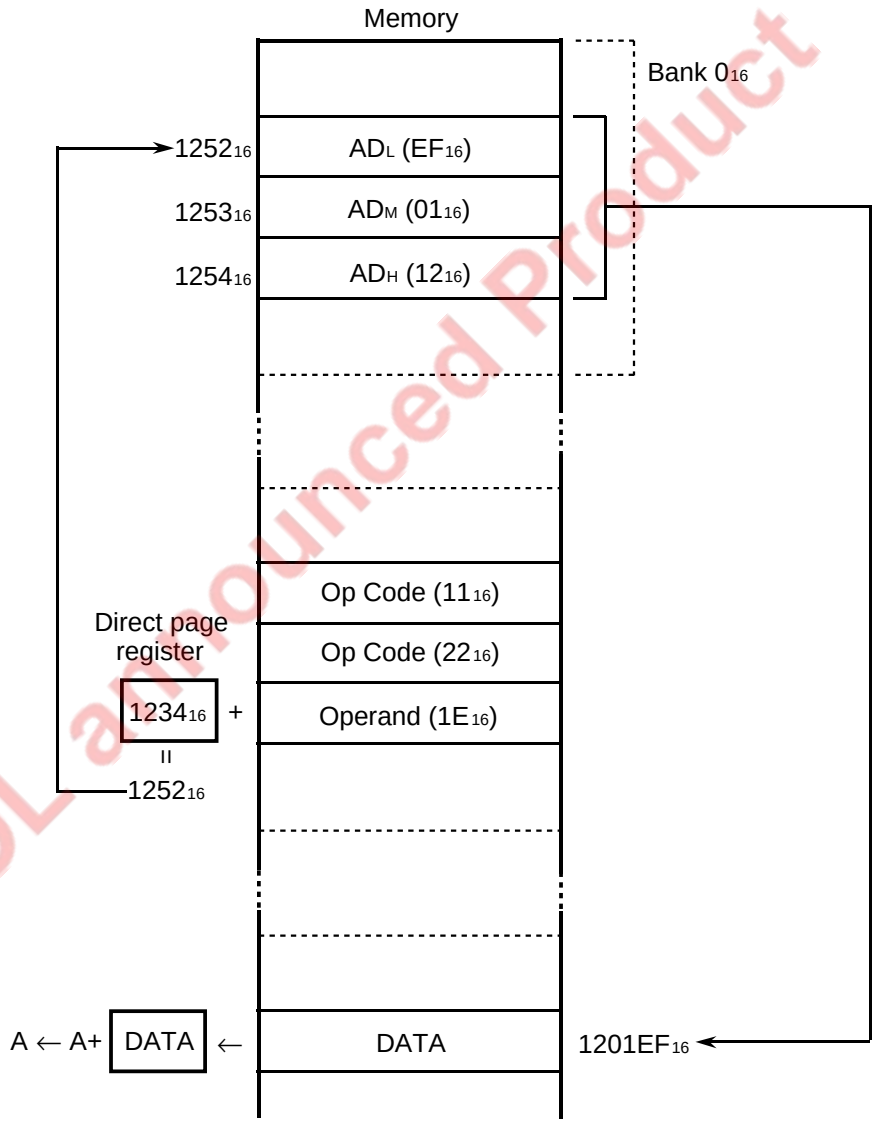
Direct Indirect Long

Mode : Direct indirect long addressing mode

Function : Specifies a sequence of 3-byte memory in bank 0_{16} by the result of adding the instruction's operand to the direct page register's contents. The contents at the address specified by the contents of these 3 bytes are an actual data. When, however, the result of adding the instruction's operand to the direct page register's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified. A sequence of 3-byte memory can cross over the bank boundary.

ex. : Mnemonic
ADD A, L (1EH)
(m="1")

Machine code
11_{16} 22_{16} $1E_{16}$

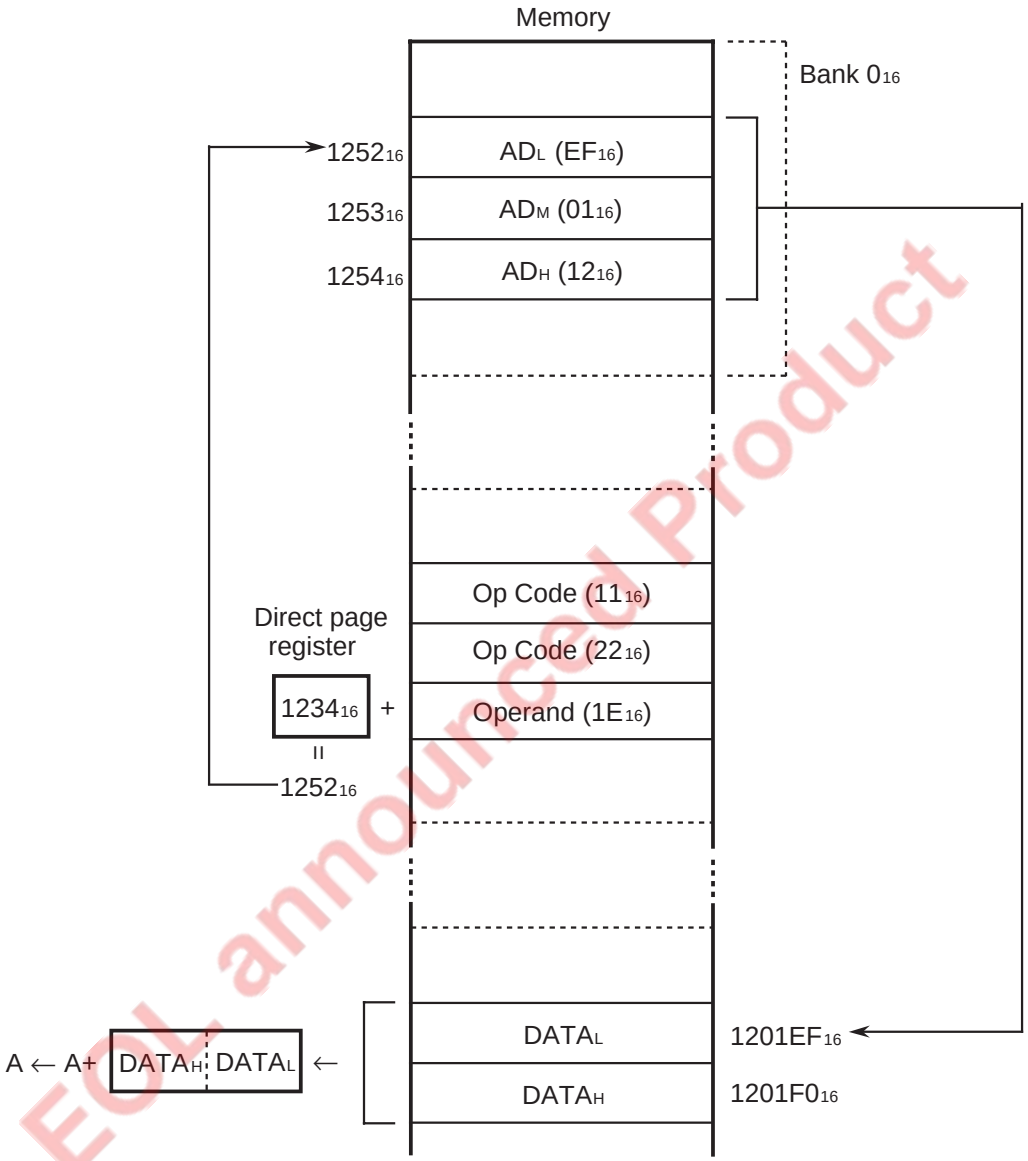


ex. : Mnemonic

ADD A, L (1EH)
(m="0")

Machine code

11_{16} 22_{16} $1E_{16}$



Direct Indirect Long Indexed Y

Mode : Direct indirect long indexed Y addressing mode

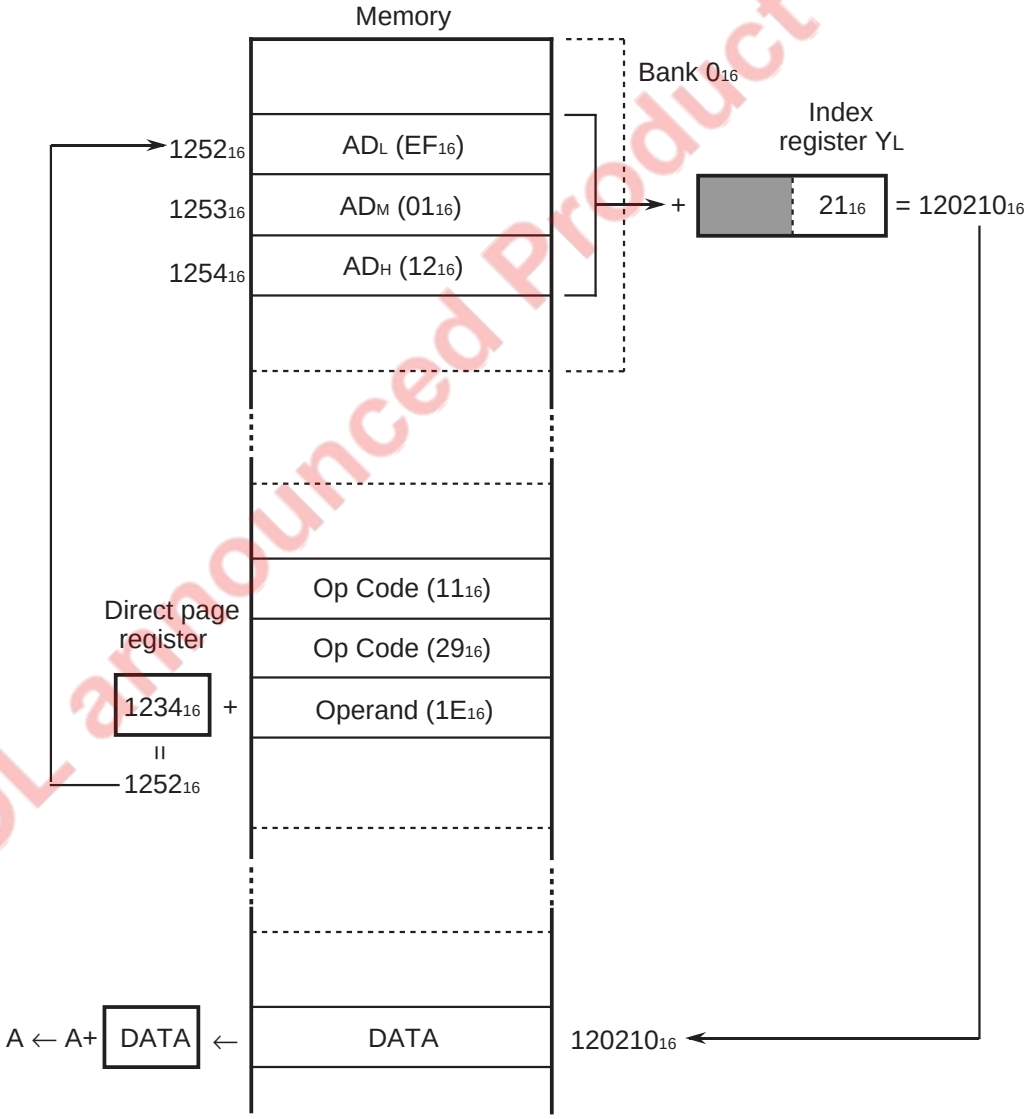
Function : Specifies a sequence of 3-byte memory in bank 0_{16} by the result of adding the instruction's operand to the direct page register's contents. The contents at the address specified by the result of adding the contents of these 3 bytes to the index register Y's contents are an actual data. When, however, the result of adding the instruction's operand to the direct page register's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified. A sequence of 3-byte memory can cross over the bank boundary.

ex. : Mnemonic

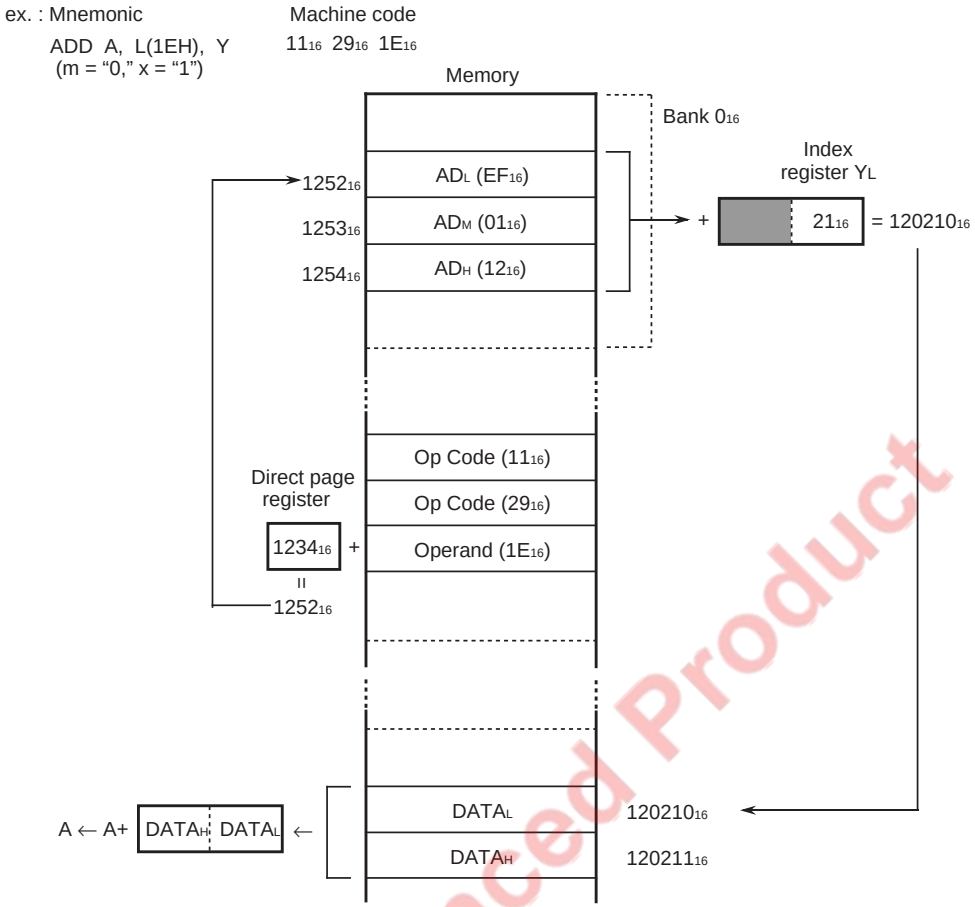
ADD A, L (1EH), Y
(m = "1," x = "1")

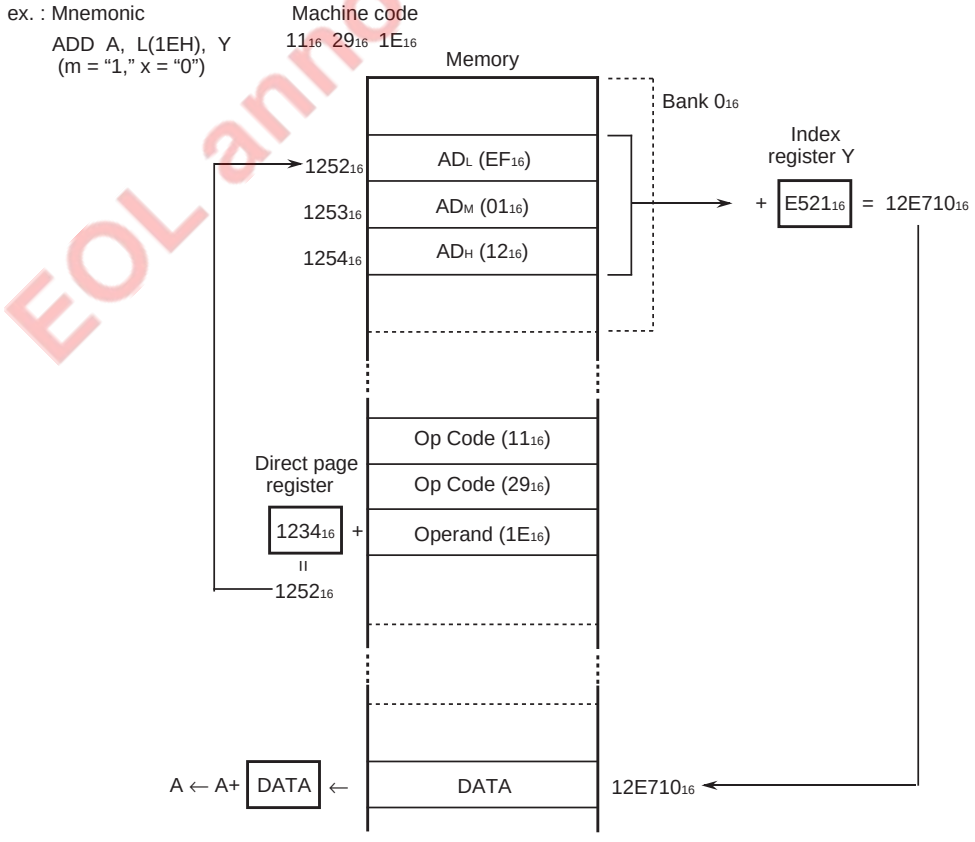
Machine code

11_{16} 29_{16} $1E_{16}$



Direct Indirect Long Indexed Y





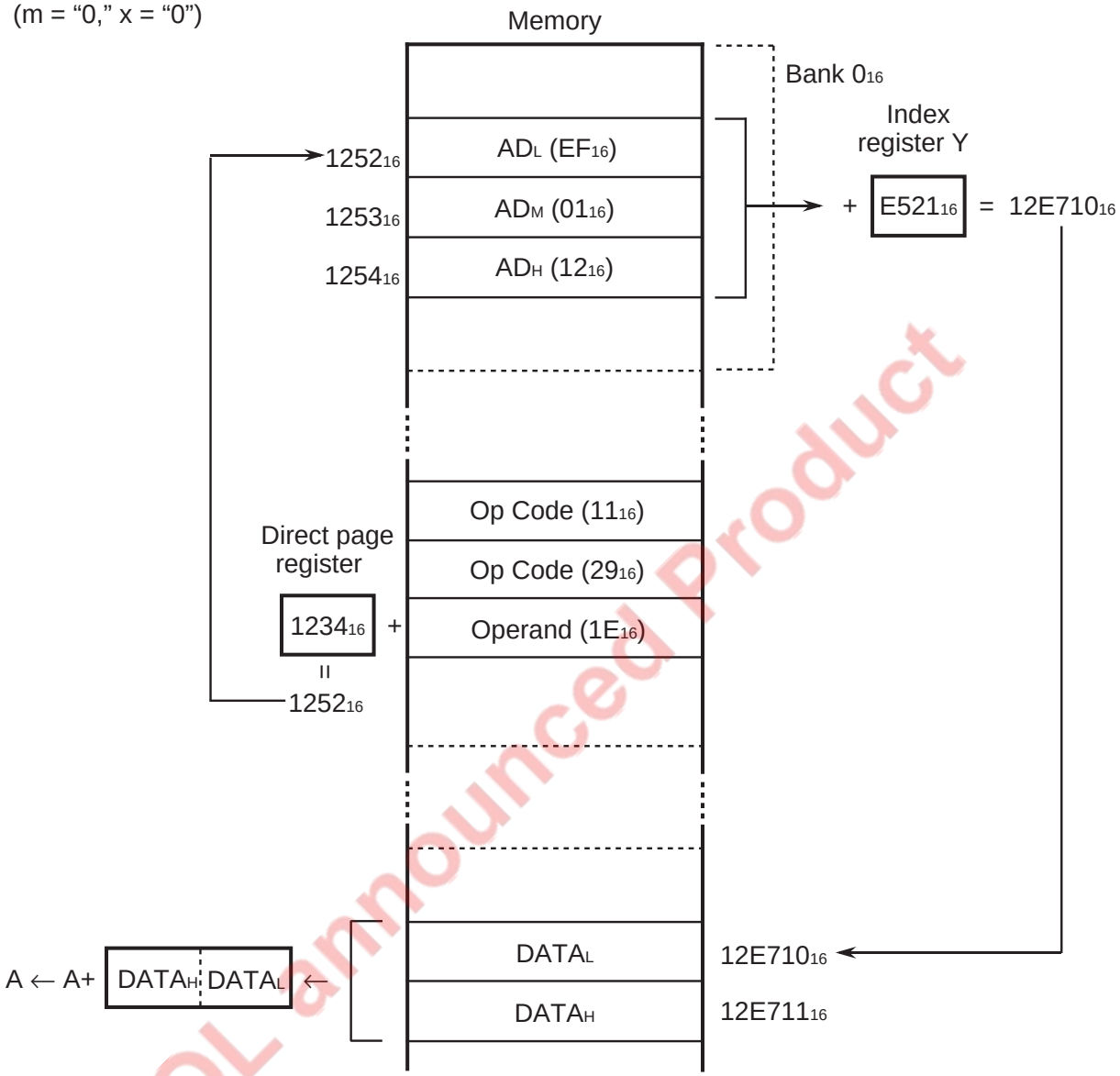
Direct Indirect Long Indexed Y

ex. : Mnemonic

ADD A, L (1EH), Y
(m = "0," x = "0")

Machine code

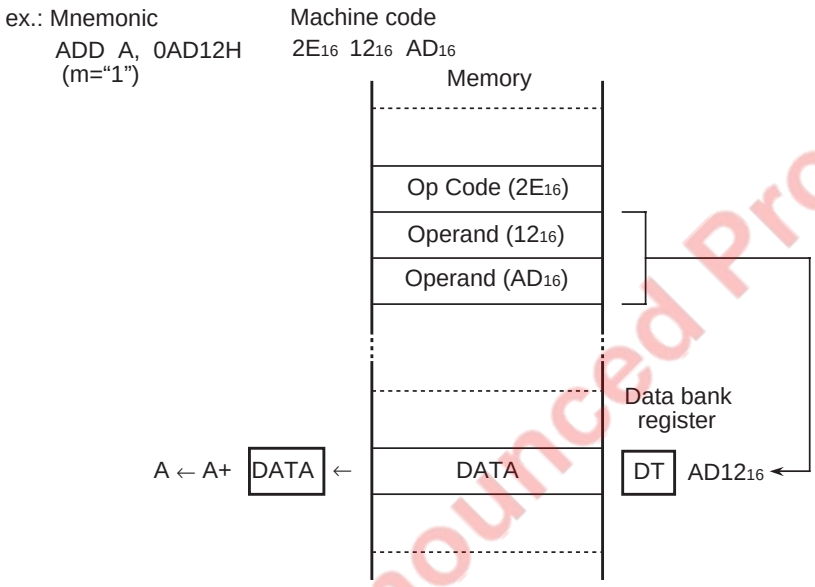
11_{16} 29_{16} $1E_{16}$

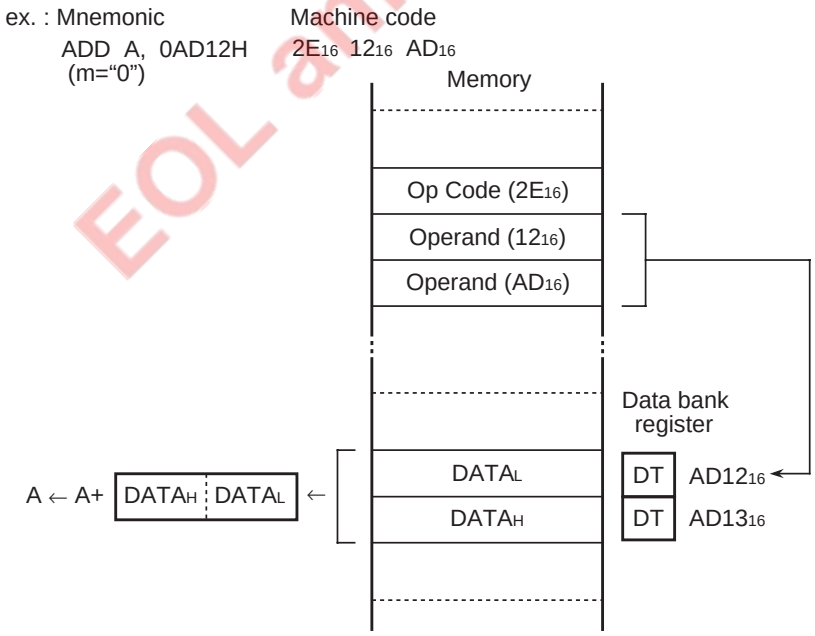


Absolute

Mode : Absolute addressing mode

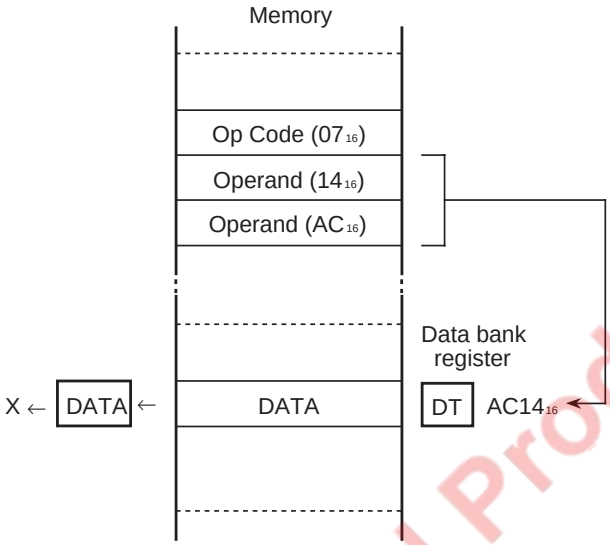
Function : The following is an actual data: the contents of the memory location specified by the instruction's operands and the contents of the data bank register. Note that, in the cases of the JMP and JSR instructions, the instruction's operands are transferred to the program counter.





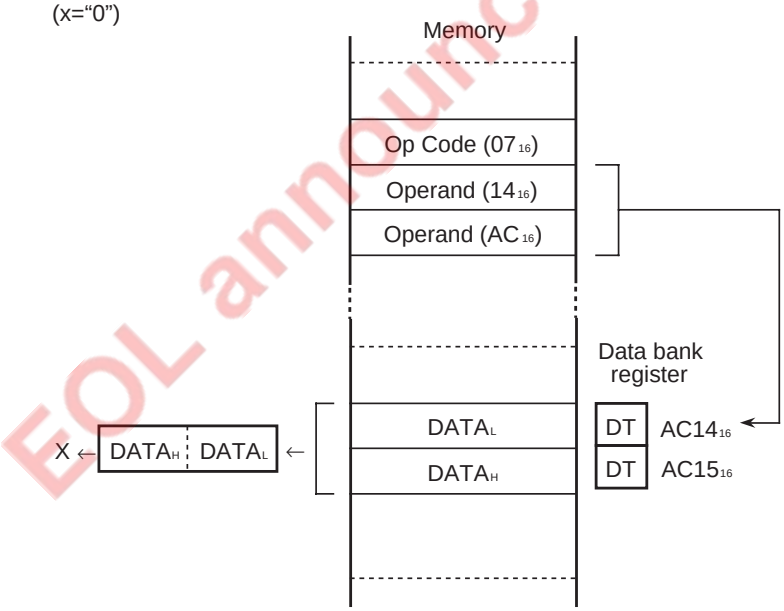
ex. : Mnemonic
LDX 0AC14H
(x="1")

Machine code
07_{16} 14_{16} AC_{16}



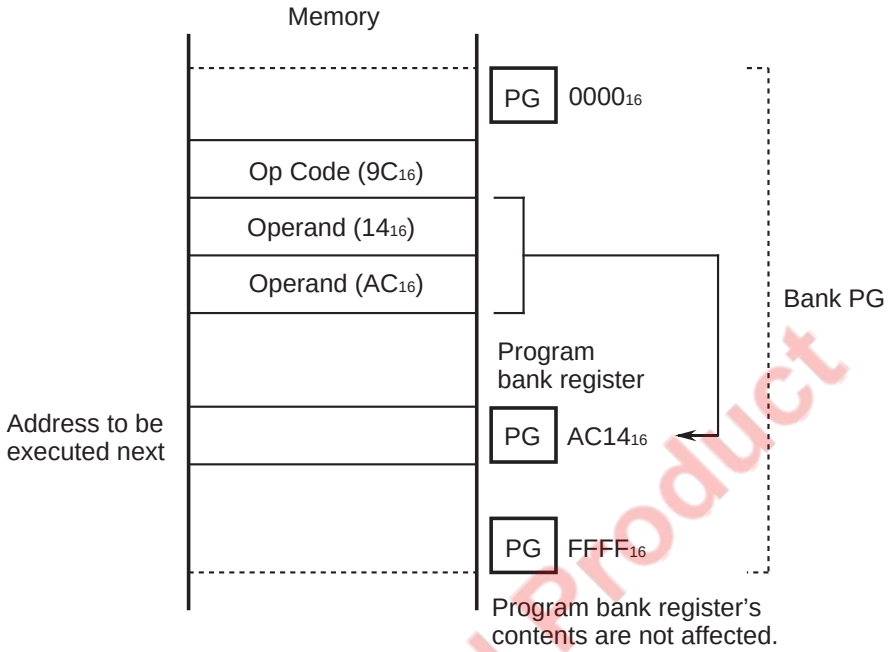
ex. : Mnemonic
LDX 0AC14H
(x="0")

Machine code
07_{16} 14_{16} AC_{16}



ex. : Mnemonic
JMP 0AC14H

Machine code
$9C_{16}$ 14_{16} AC_{16}

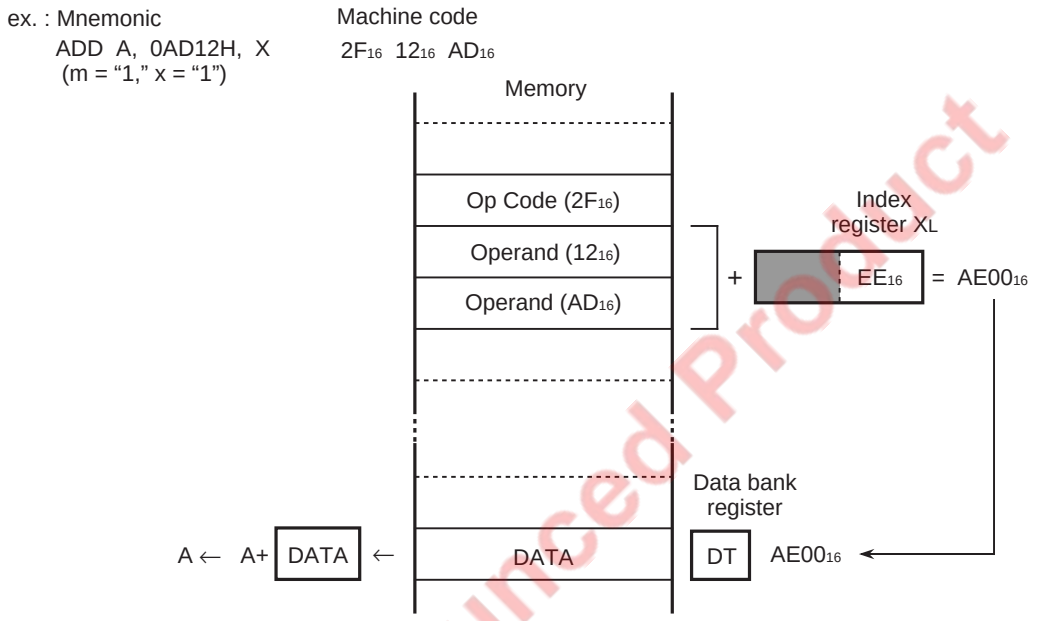


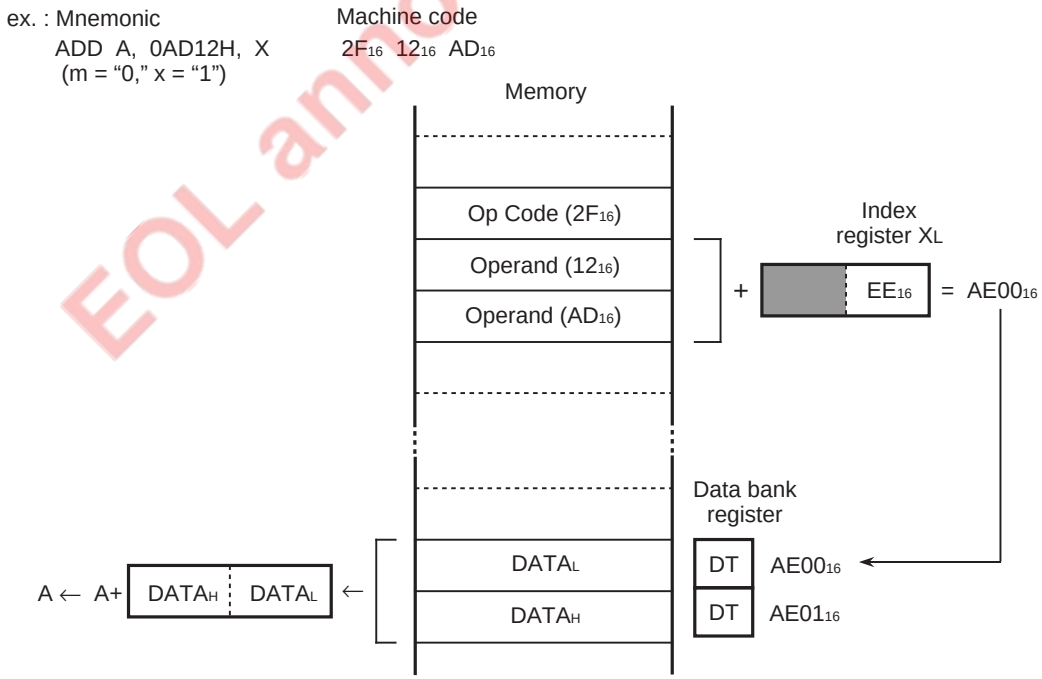
Note : Note the branch destination bank in the case where a JMP or a JSR instruction is located near a bank boundary.
⇒Refer to the description of a JMP/JMPL instruction (Page 4-111).
Refer to the description of a JSR/JSRL instruction (Page 4-112).

Absolute Indexed X

Mode : Absolute indexed X addressing mode

Function : The following is an actual data: the contents of the memory location specified by the result of adding a 16-bit length numerical value expressed with the instruction's operands to the index register X's contents, and the contents of the data bank register. If, however, the addition of the numerical value expressed with the instruction's operands and the index register X's contents generates a carry, the value which is 1 larger than the contents of the data bank register indicates the bank.

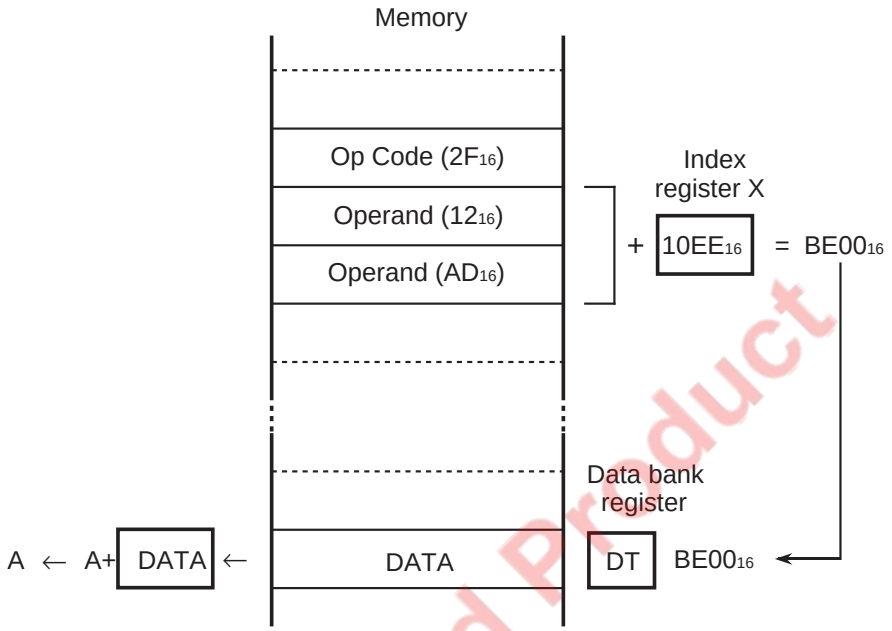




Absolute Indexed X

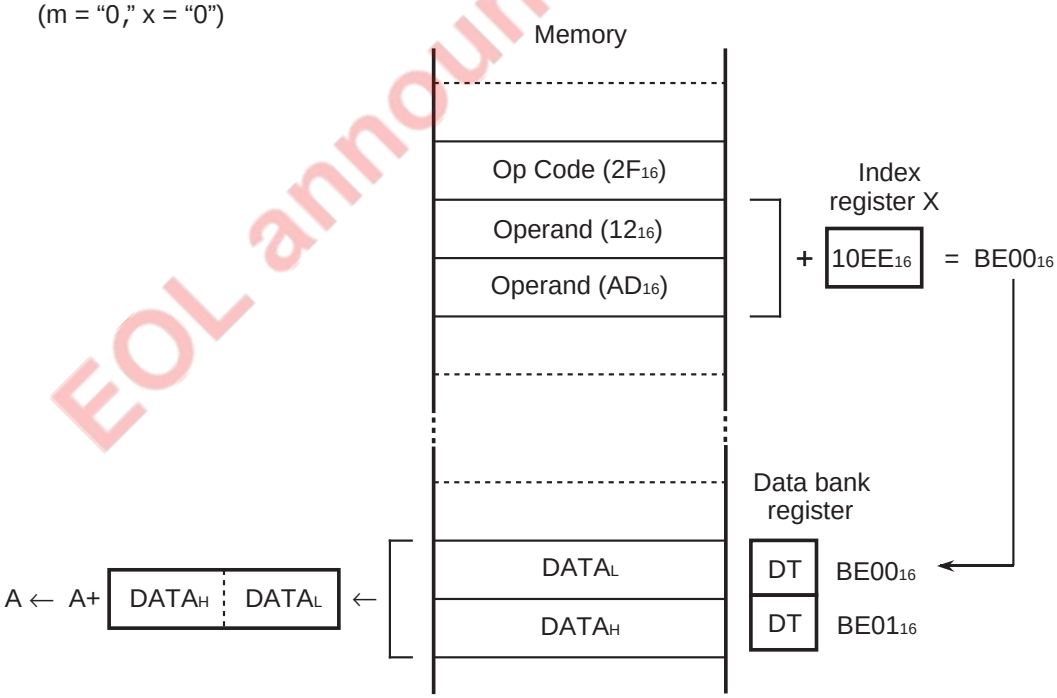
ex. : Mnemonic
ADD A, 0AD12H, X
(m = "1," x = "0")

Machine code
$2F_{16}$ 12_{16} AD_{16}



ex. : Mnemonic
ADD A, 0AD12H, X
(m = "0," x = "0")

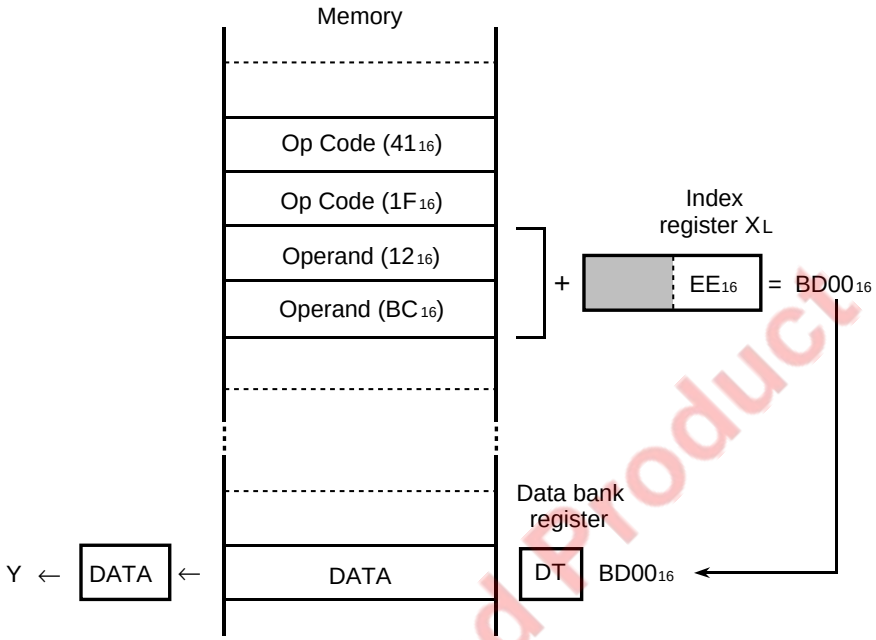
Machine code
$2F_{16}$ 12_{16} AD_{16}



Absolute Indexed X

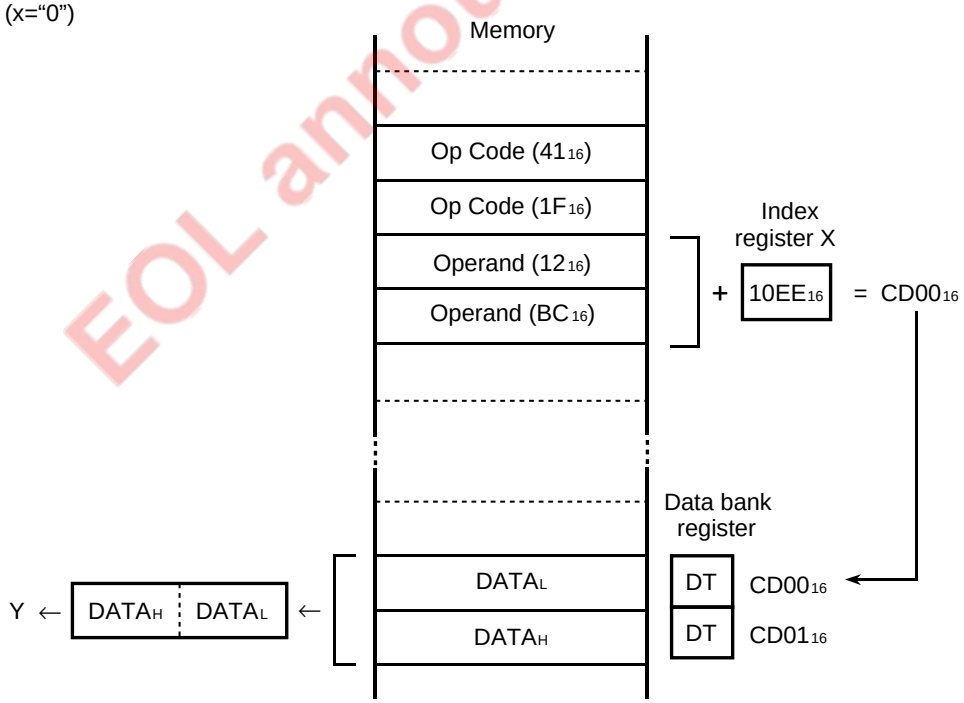
ex. : Mnemonic
LDY 0BC12H, X
(x="1")

Machine code
41_{16} $1F_{16}$ 12_{16} BC_{16}



ex. : Mnemonic
LDY 0BC12H, X
(x="0")

Machine code
41_{16} $1F_{16}$ 12_{16} BC_{16}



Absolute Indexed Y

Mode : Absolute indexed Y addressing mode

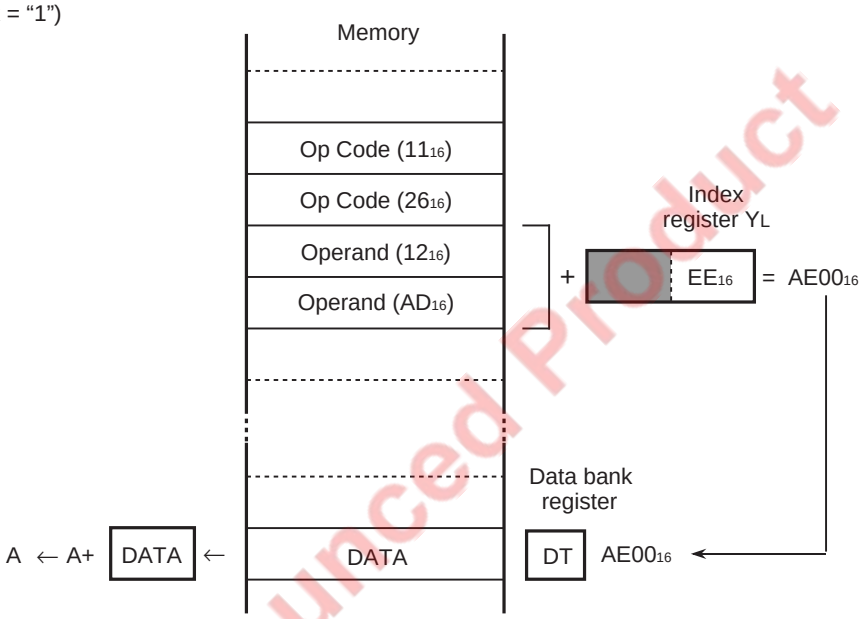
Function : The following is an actual data: the contents of the memory location specified by the result of adding a 16-bit length numerical value expressed with the instruction's operands to the index register Y's contents, and the contents of the data bank register. If, however, the addition of the numerical value expressed with the instruction's operands to the index register Y's contents generates a carry, the value which is 1 larger than the contents of the data bank register indicates the bank.

ex. : Mnemonic

ADD A, 0AD12H, Y
(m = "1," x = "1")

Machine code

11_{16} 26_{16} 12_{16} AD_{16}

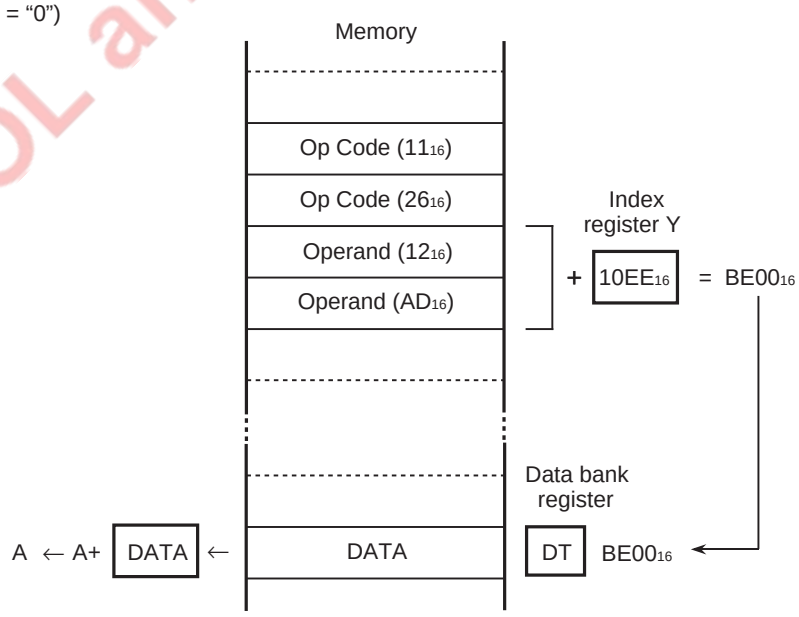


ex. : Mnemonic

ADD A, 0AD12H, Y
(m = "1," x = "0")

Machine code

11_{16} 26_{16} 12_{16} AD_{16}



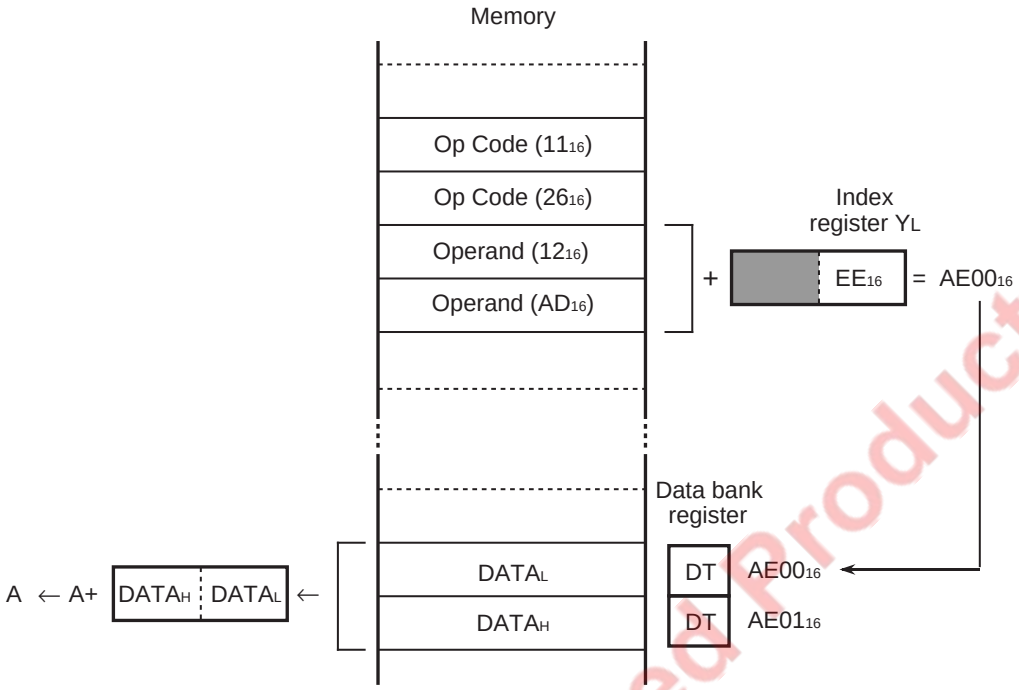
Absolute Indexed Y

ex. : Mnemonic

ADD A, 0AD12H, Y
(m = "0," x = "1")

Machine code

11_{16} 26_{16} 12_{16} AD_{16}

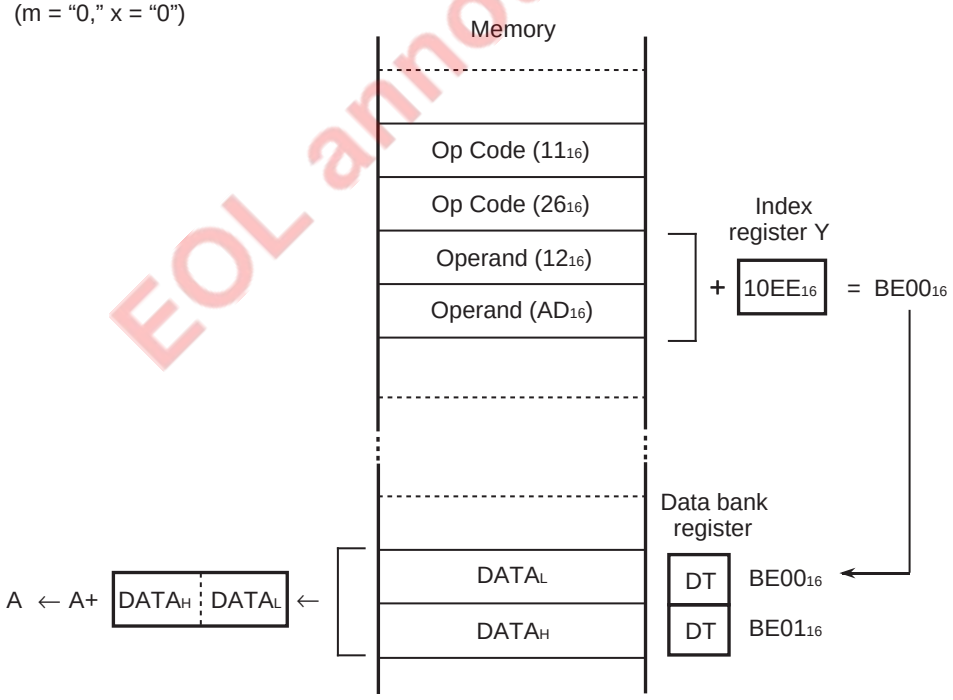


ex. : Mnemonic

ADD A, 0AD12H, Y
(m = "0," x = "0")

Machine code

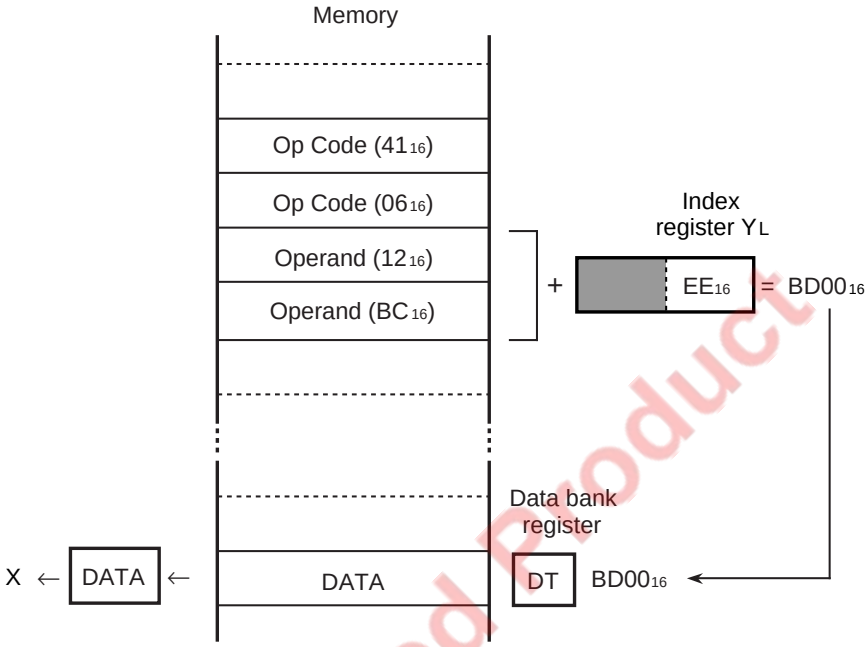
11_{16} 26_{16} 12_{16} AD_{16}



Absolute Indexed Y

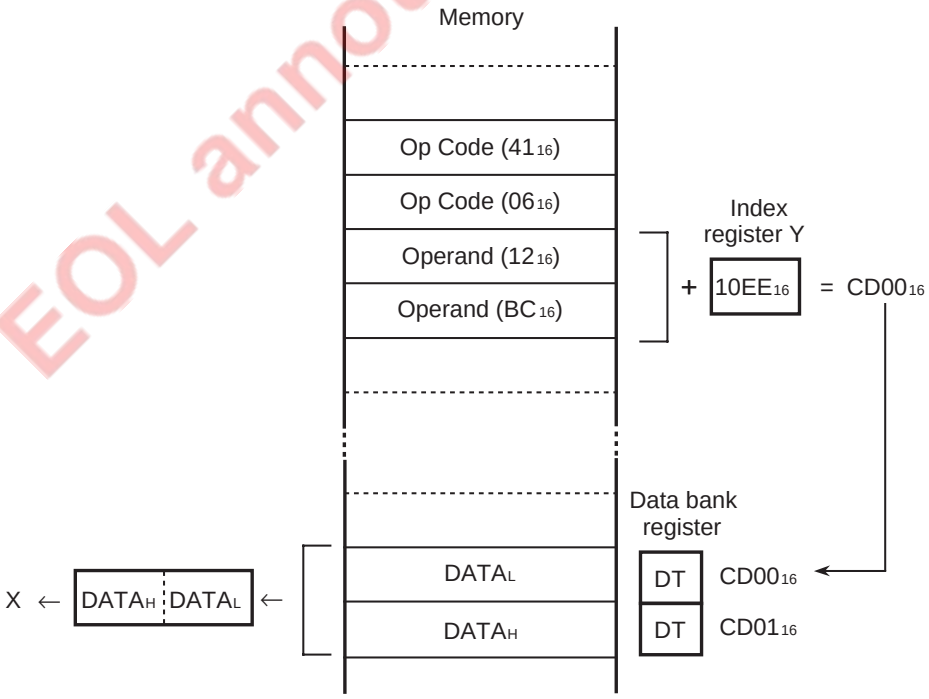
ex. : Mnemonic
LDX 0BC12H, Y
(x="1")

Machine code
41_{16} 06_{16} 12_{16} BC_{16}



ex. : Mnemonic
LDX 0BC12H, Y
(x="0")

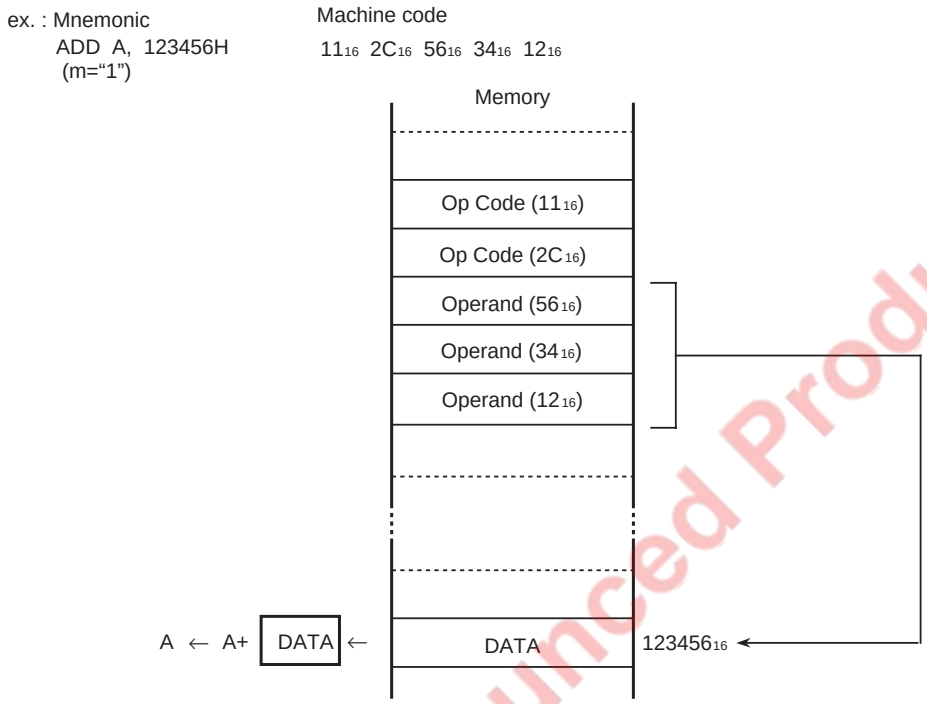
Machine code
41_{16} 06_{16} 12_{16} BC_{16}

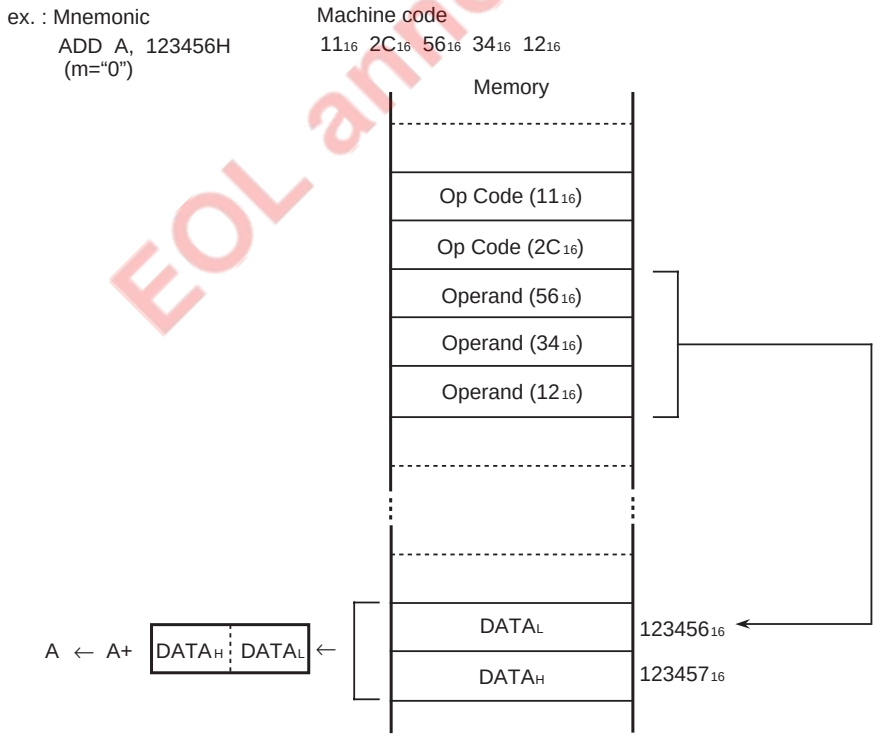


Absolute Long

Mode : Absolute long addressing mode

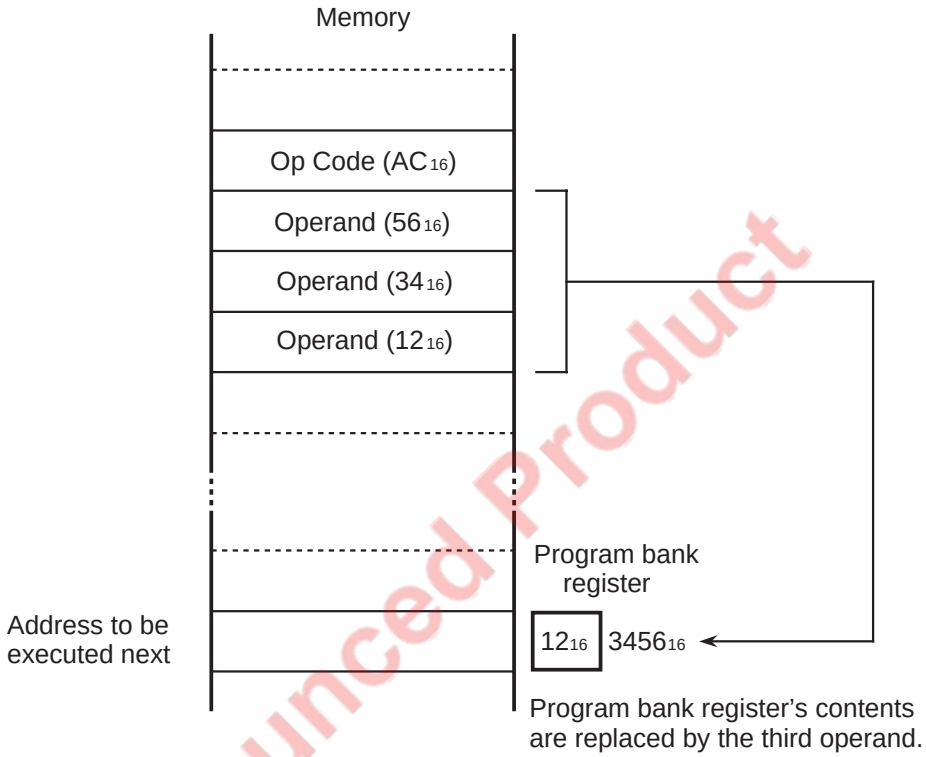
Function : The contents of the memory location specified by the instruction's operands are an actual data. Note that, in the cases of the JMPL and JSRL instructions, the instruction's second and third bytes are transferred to the program counter and the fourth byte is transferred to the program bank register.





ex. : Mnemonic
JMPL 123456H

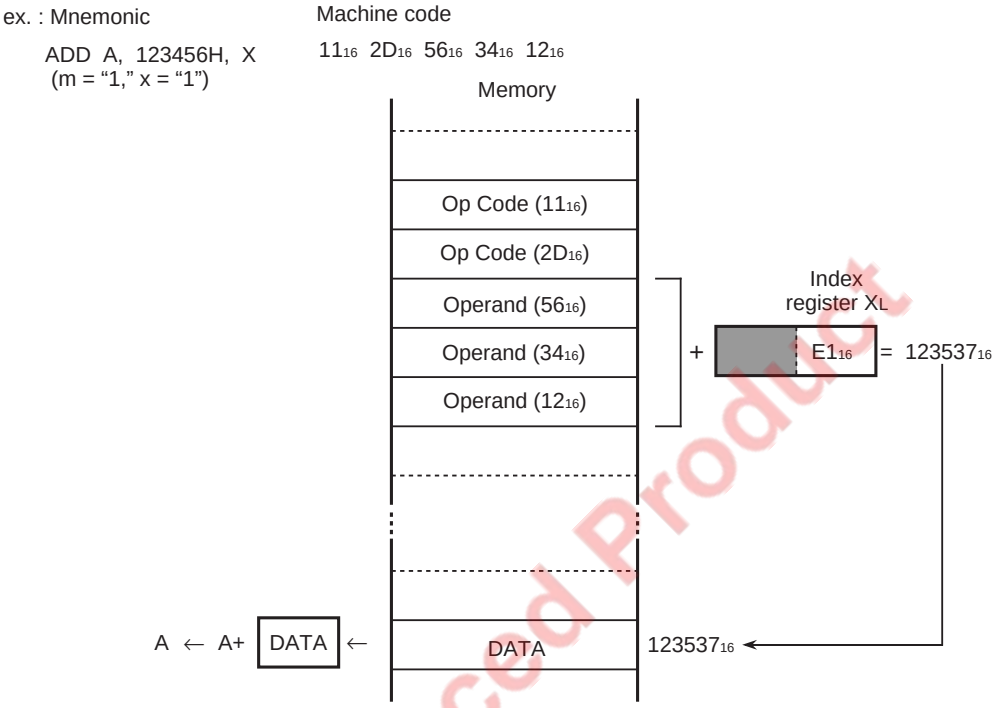
Machine code
AC_{16} 56_{16} 34_{16} 12_{16}

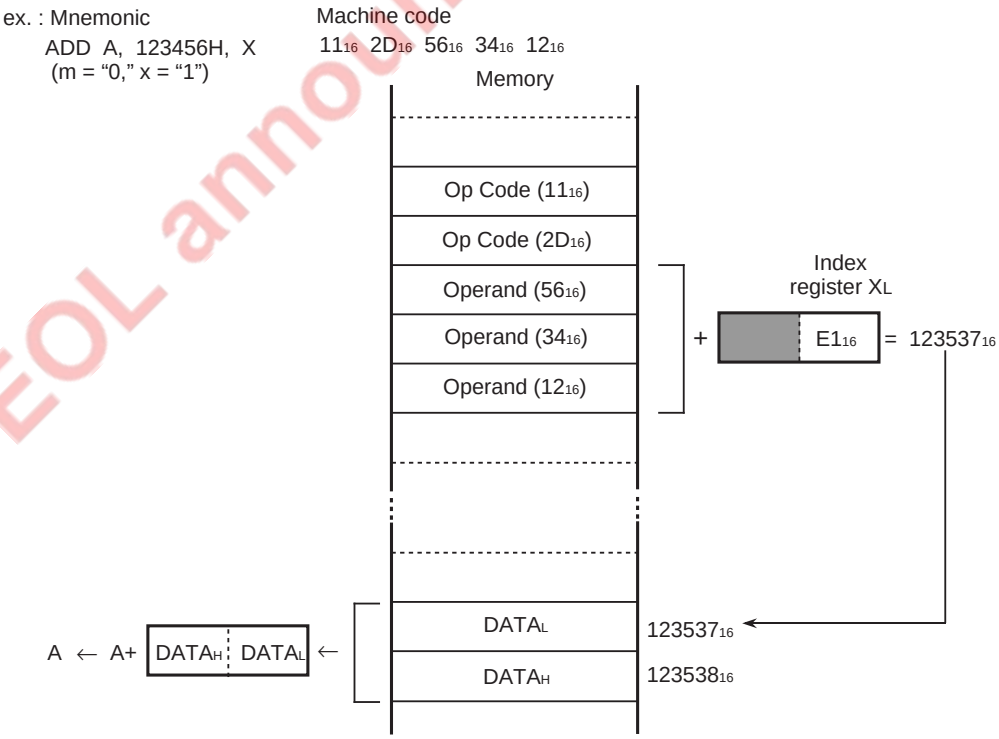


Absolute Long Indexed X

Mode : Absolute long indexed X addressing mode

Function : The following is an actual data: the contents of the memory location specified by the result of adding a numerical value expressed with the instruction's operands to the index register X's contents.





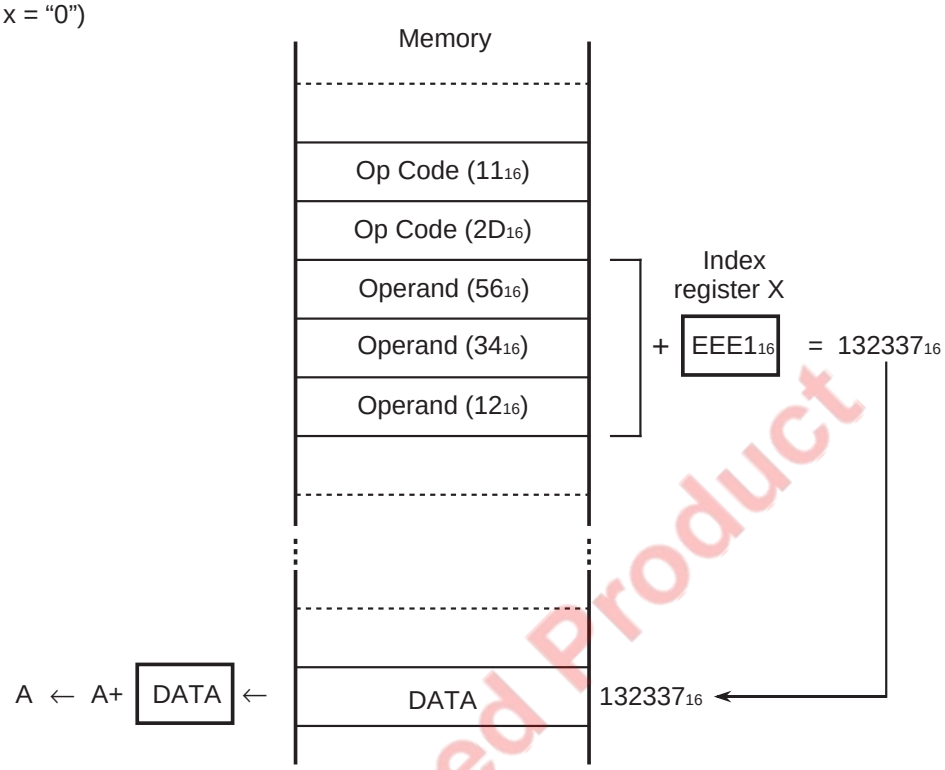
Absolute Long Indexed X

ex. : Mnemonic

ADD A, 123456H, X
(m = "1," x = "0")

Machine code

11_{16} $2D_{16}$ 56_{16} 34_{16} 12_{16}

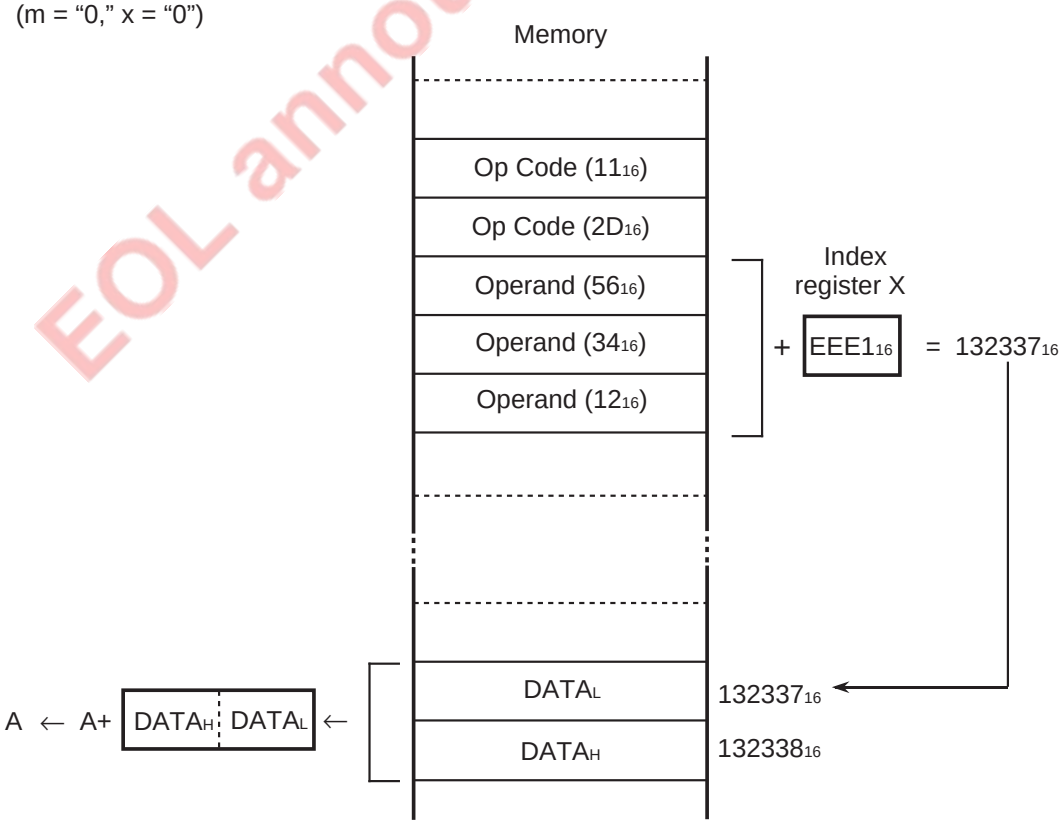


ex. : Mnemonic

ADD A, 123456H, X
(m = "0," x = "0")

Machine code

11_{16} $2D_{16}$ 56_{16} 34_{16} 12_{16}



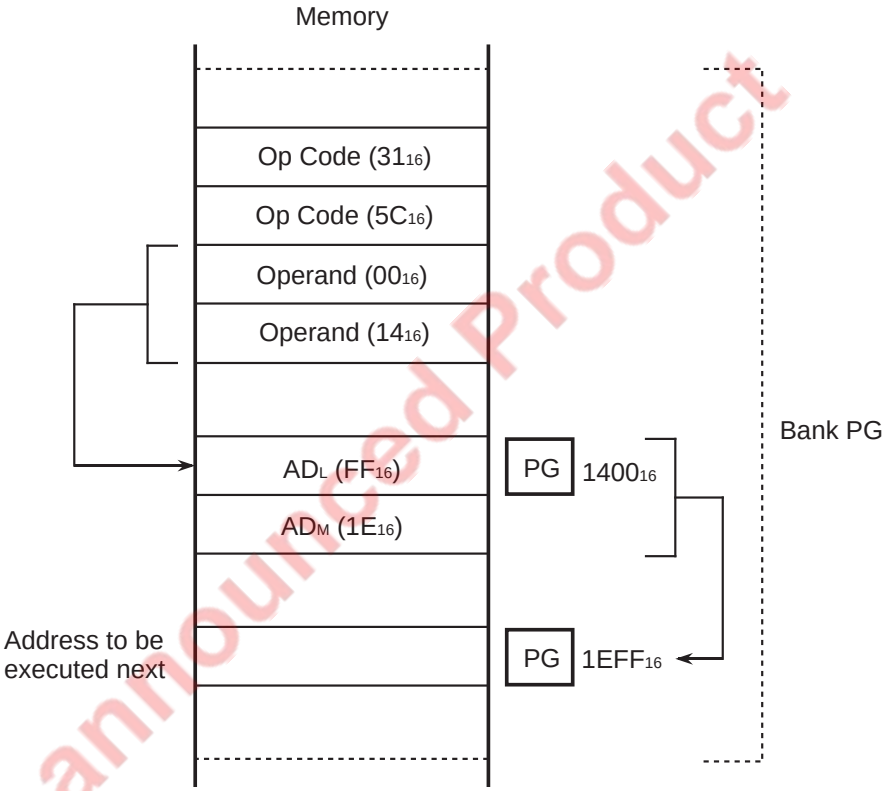
Absolute Indirect

Mode : Absolute indirect addressing mode

Function : A sequence of 2-byte memory is specified by the instruction's third and fourth bytes in the same program bank. The contents of this 2-byte memory specify the branch destination address within the same program bank.
This addressing mode is used by a JMP instruction.

ex. : Mnemonic
JMP (1400H)

Machine code
31_{16} $5C_{16}$ 00_{16} 14_{16}



Note : Note the reference/branch destination bank when an instruction or a reference destination is located near a bank boundary.
⇒ Refer to the description of a JMP/JMPL instruction (Page 4-111).

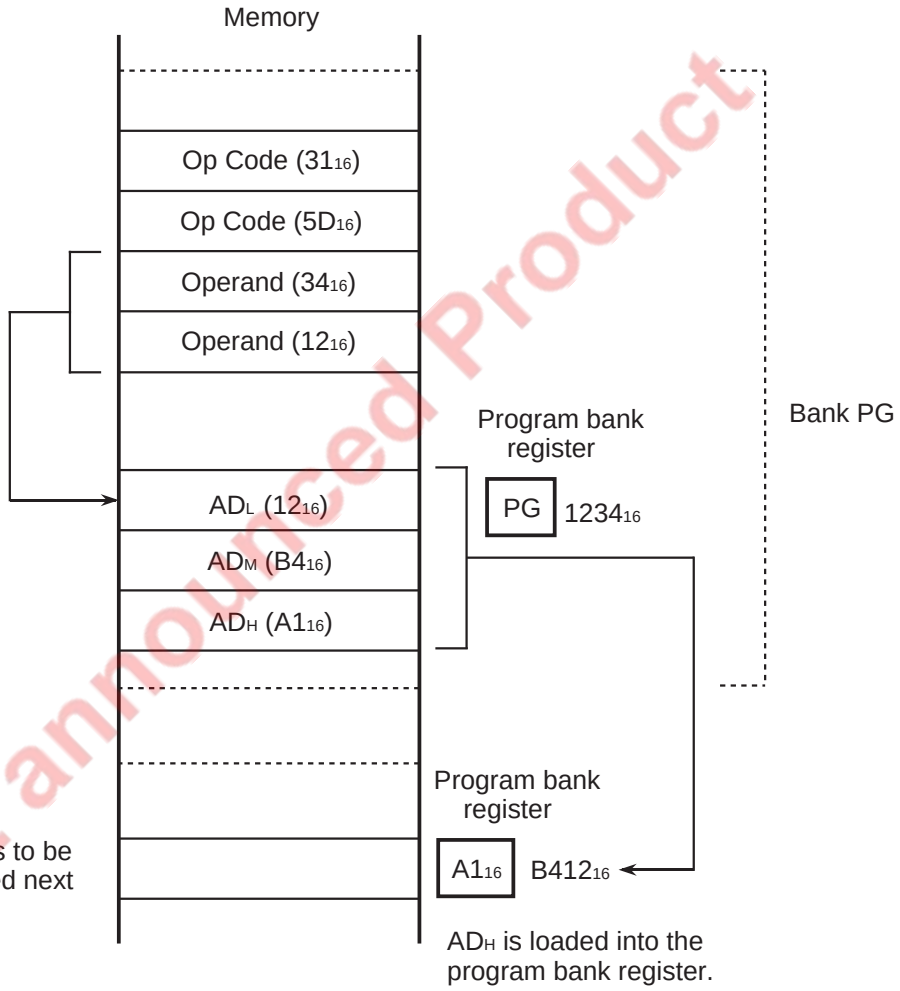
Absolute Indirect Long

Mode : Absolute indirect long addressing mode

Function : A sequence of 3-byte memory is specified by the instruction's third and fourth bytes in the same program bank. The contents of this 3-byte memory specify the branch destination address. This addressing mode is used by a JMPL instruction.

ex. : Mnemonic
JMPL L(1234H)

Machine code
31_{16} $5D_{16}$ 34_{16} 12_{16}



Note : Note the reference destination bank when an instruction is located near a bank boundary.
⇒Refer to the description of a JMP/JMPL instruction (Page 4-111).

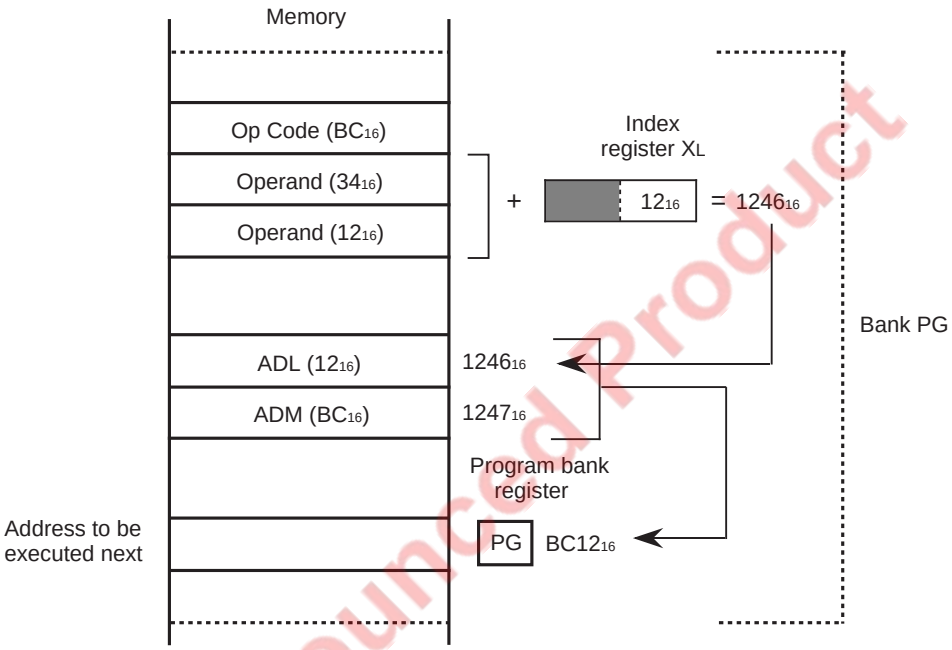
Absolute Indexed X Indirect

Mode : Absolute indexed X indirect addressing mode

Function : A sequence of 2-byte memory is specified by the result of adding a numerical value expressed with the instruction's second and third bytes to the index register X's contents; the memory bank is specified by program bank register PG at this time. The contents of this 2-byte memory specify the branch destination address.
This addressing mode is used by a JMP and a JSR instructions.

ex.: Mnemonic
JMP (1234H, X)
(x = "1")

Machine code
BC_{16} 34_{16} 12_{16}



Note : Note the reference/branch destination bank in the case of a JMP or a JSR instruction when the instruction or the branch destination address is located near a bank boundary.

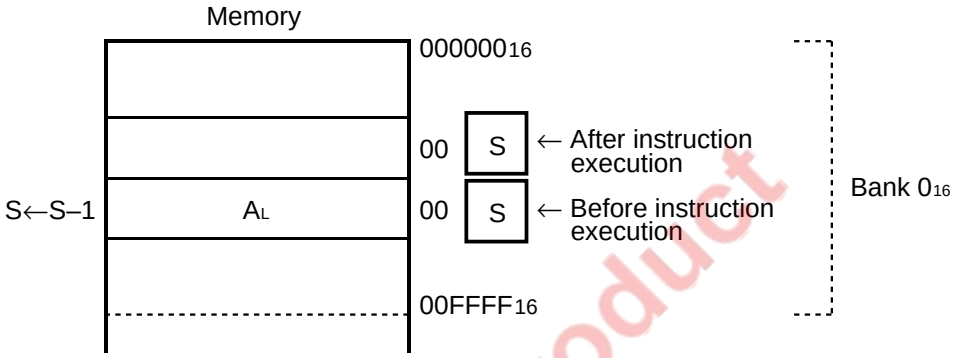
➢ Refer to the description of a JMP/JMPL instruction (Page 4-111).
Refer to the description of a JSR/JSRL instruction (Page 4-112).

Mode : Stack addressing mode

Function : The contents of a register or others are stored to or restored from the memory of which location is specified by the stack pointer; this memory is called “stack area.” The stack area is set in bank 0_{16}.

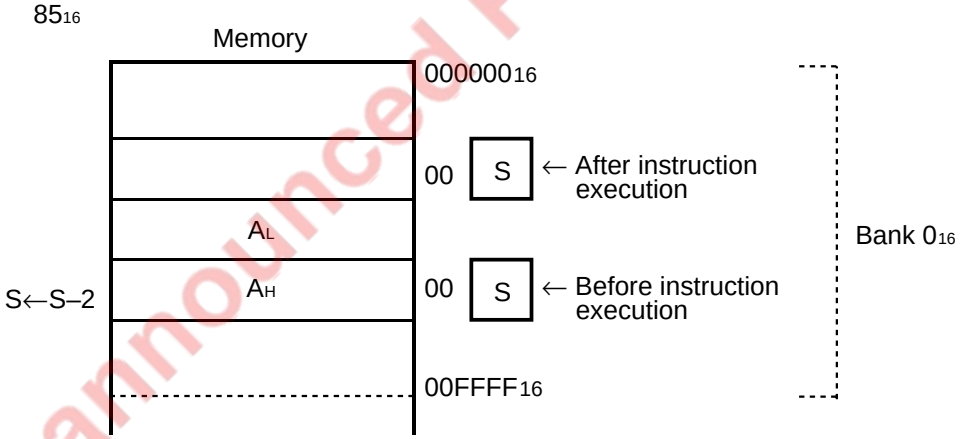
ex. : Mnemonic
PHA
(m=“1”)

Machine code
85_{16}



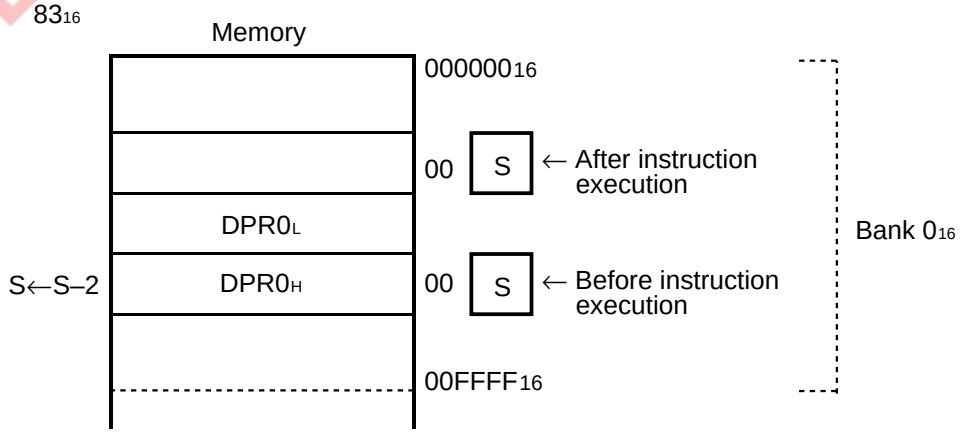
ex. : Mnemonic
PHA
(m=“0”)

Machine code
85_{16}



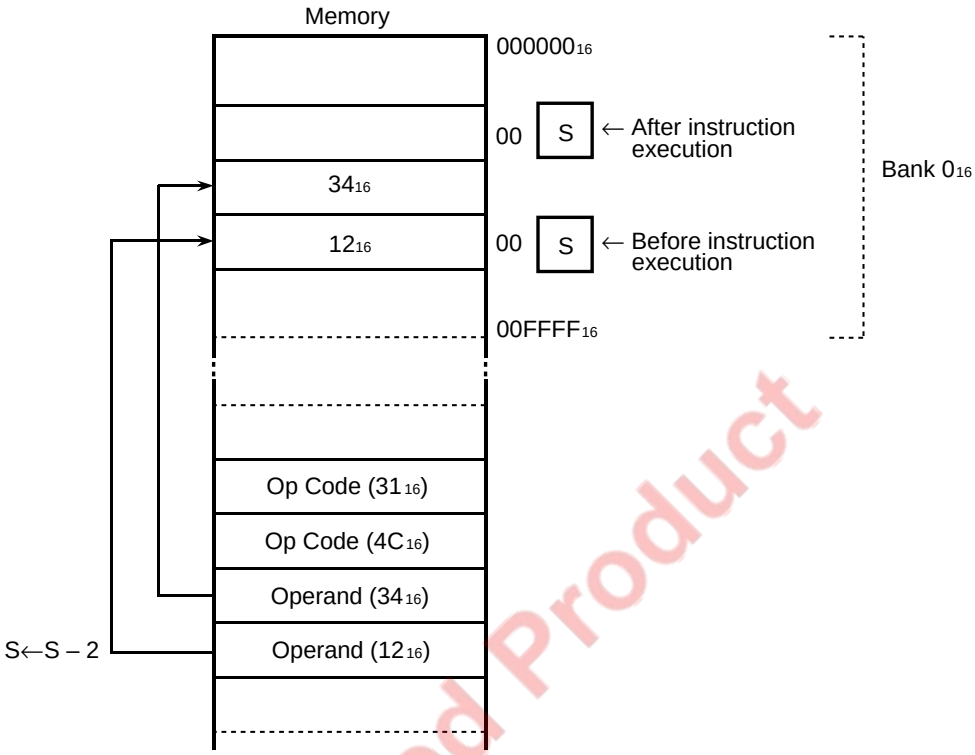
ex. : Mnemonic
PHD

Machine code
83_{16}



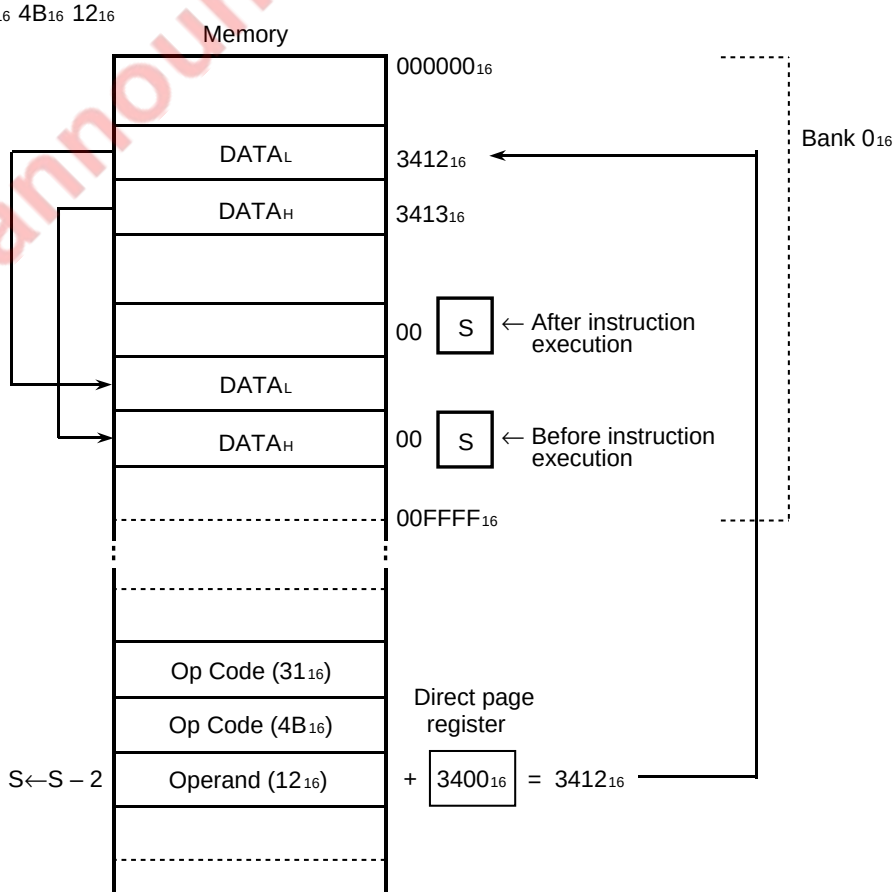
ex. : Mnemonic
PEA #1234H

Machine code
31_{16} $4C_{16}$ 34_{16} 12_{16}



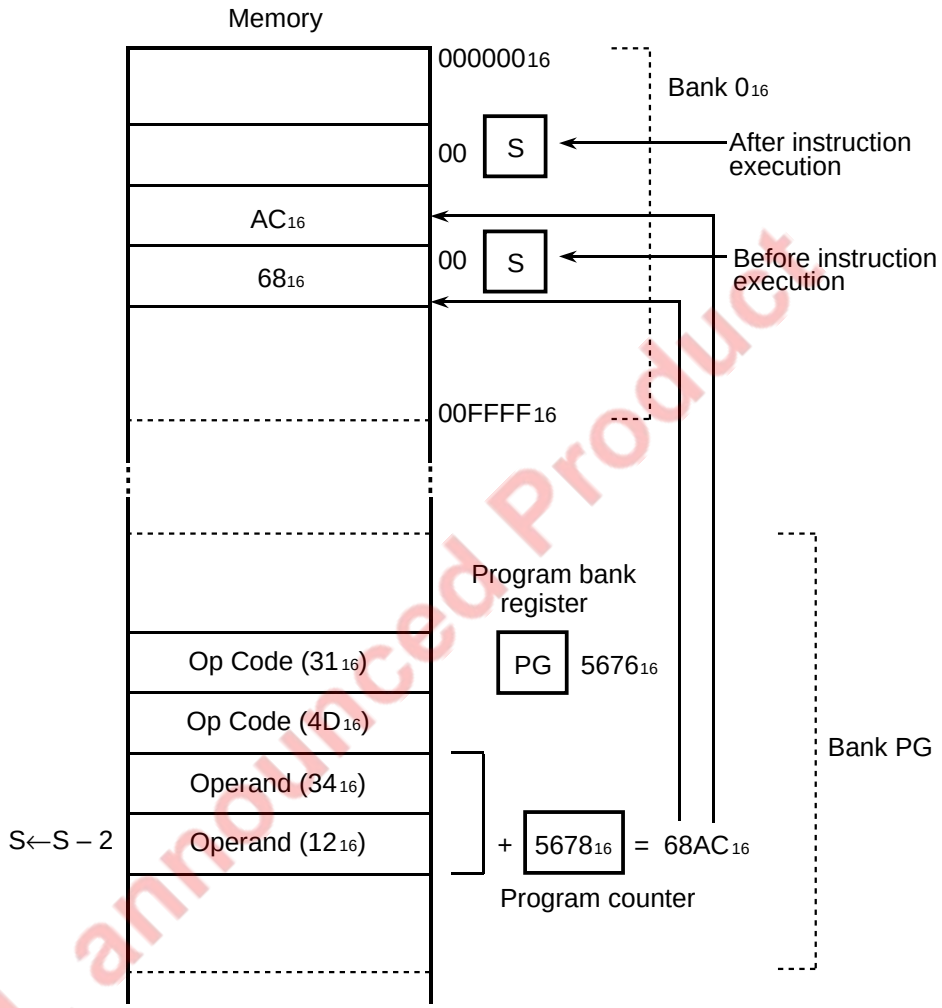
ex. : Mnemonic
PEI 12H

Machine code
31_{16} $4B_{16}$ 12_{16}



ex. : Mnemonic
PER #1234H

Machine code
31_{16} $4D_{16}$ 34_{16} 12_{16}



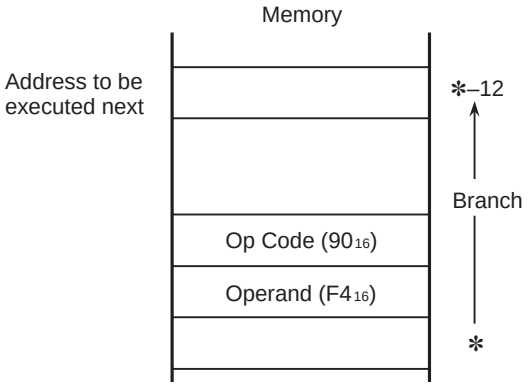
Relative

Mode : Relative addressing mode

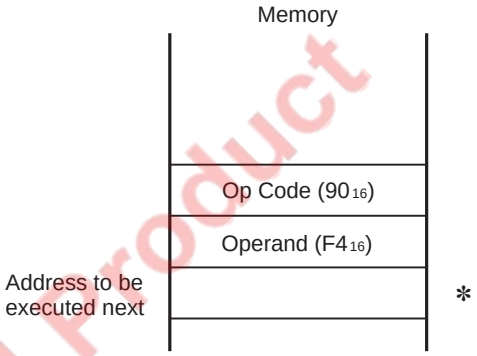
Function : Branches to the address specified by the result of adding the program counter's contents to the instruction's second byte. In the case of a long branch with the BRA instruction, the instruction's second and third bytes are added to the program counter's contents as a 15-bit signed numerical value. In the case of the BSR instruction, the instruction's 3 bits of the first byte and the second byte are added to the program counter's contents as a 11-dit signed numerical value. If the addition generates a carry or a borrow, 1 is added to or subtracted from the program bank register.

ex. : Mnemonic
BCC ✻ – 12

Machine code
90_{16} $F4_{16}$



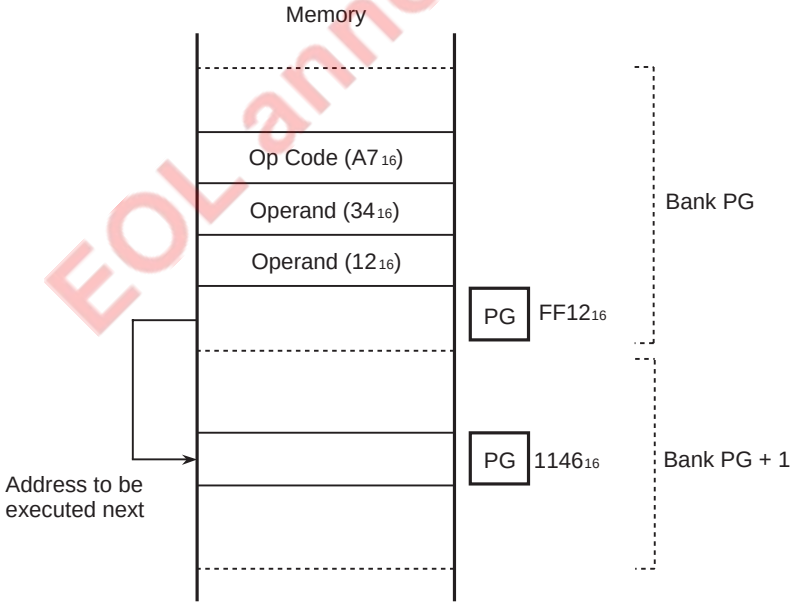
Branches to the address ✻ – 12 when the carry flag (C) is "0."



Advances to the address ✻ when the carry flag (C) is "1."

ex. : Mnemonic
BRAL 1234H

Machine code
$A7_{16}$ 34_{16} 12_{16}



Direct Bit Relative

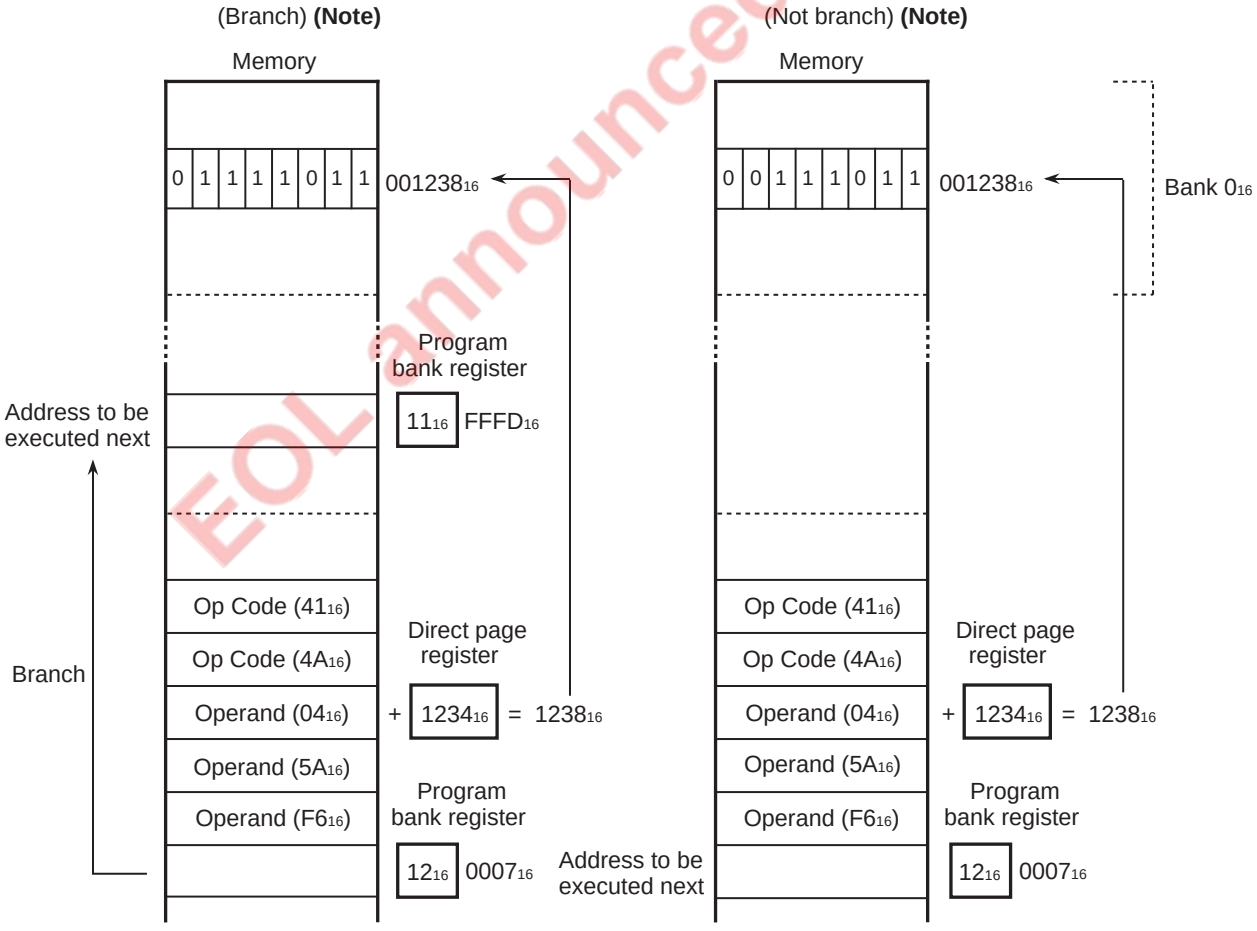
Mode : Direct bit relative addressing mode

Function :

- BBC and BBS instructions
  Specifies the memory location in bank 0_{16} by the result of adding the instruction's third byte to the direct page register's contents; specifies the multiple bits' position in that memory by the bit pattern of the instruction's fourth and fifth bytes (when the m flag is "1," the fourth byte only). Then, when the specified bits all satisfy the branching conditions, branches to the address specified by the result of adding the instruction's sixth byte (or when the m flag is "1," the fifth byte) as a signed numerical value to the program counter's contents. When, however, the result of adding the instruction's second byte to the direct page register's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified.
- BBCB and BBSB instructions
  Specifies the memory location in bank 0_{16} by the result of adding the instruction's second byte to the direct page register's contents; specifies the multiple bits' position in that memory by the bit pattern of the instruction's third byte. Then, when the specified bits all satisfy the branching conditions, branches to the address specified by the result of adding the instruction's fourth byte as a signed numerical value to the program counter's contents. When, however, the result of adding the instruction's second byte to the direct page register's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified.

ex. : Mnemonic
BBS #5AH, 04H, 0F6H
(m = "1")

Machine code
41_{16} $4A_{16}$ 04_{16} $5A_{16}$ $F6_{16}$



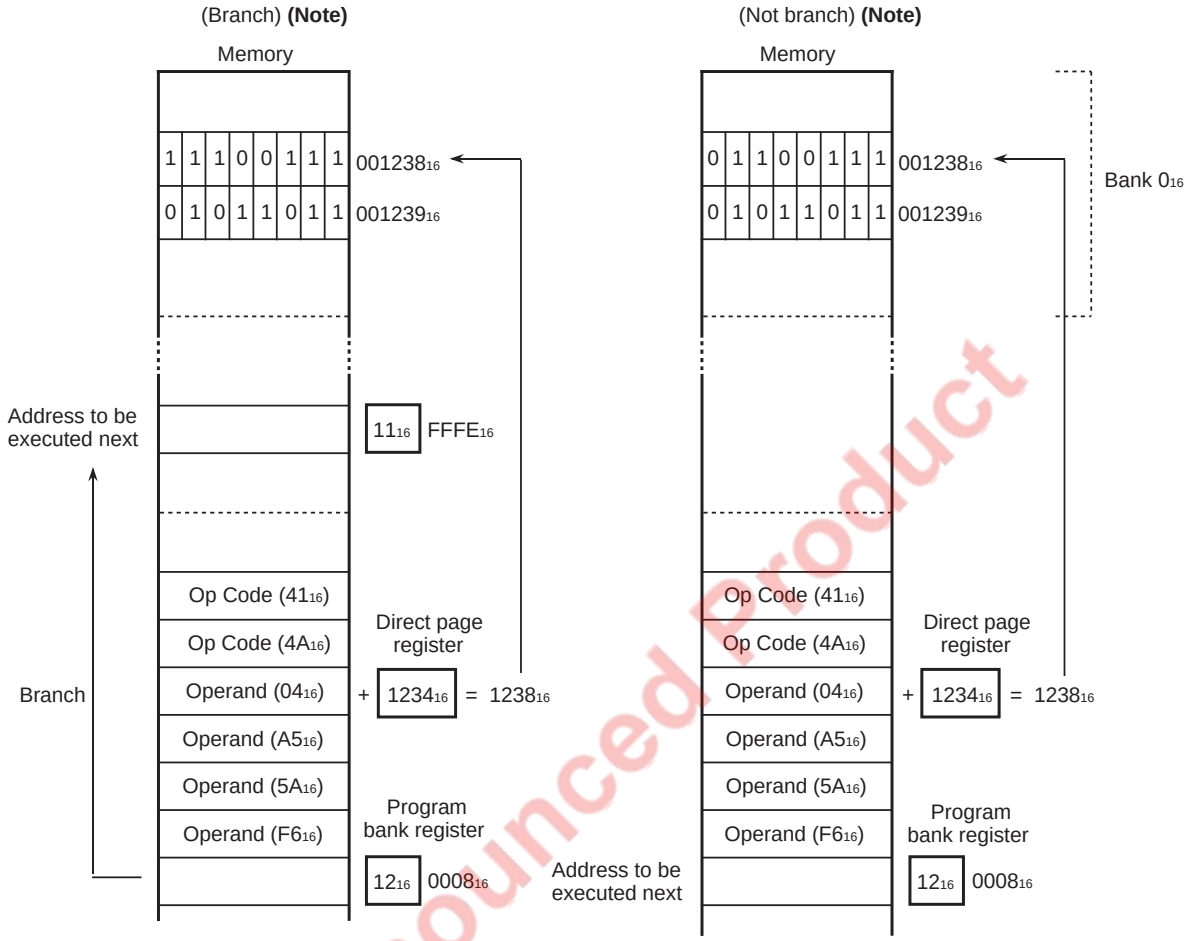
Note: Whether to branch or not depends on the branching conditions.

ex. : Mnemonic

BBS #5AA5H, 04H, 0F6H
(m = "0")

Machine code

41_{16} $4A_{16}$ 04_{16} $A5_{16}$ $5A_{16}$ $F6_{16}$



Note: Whether to branch or not depends on the branching conditions.

Absolute Bit Relative

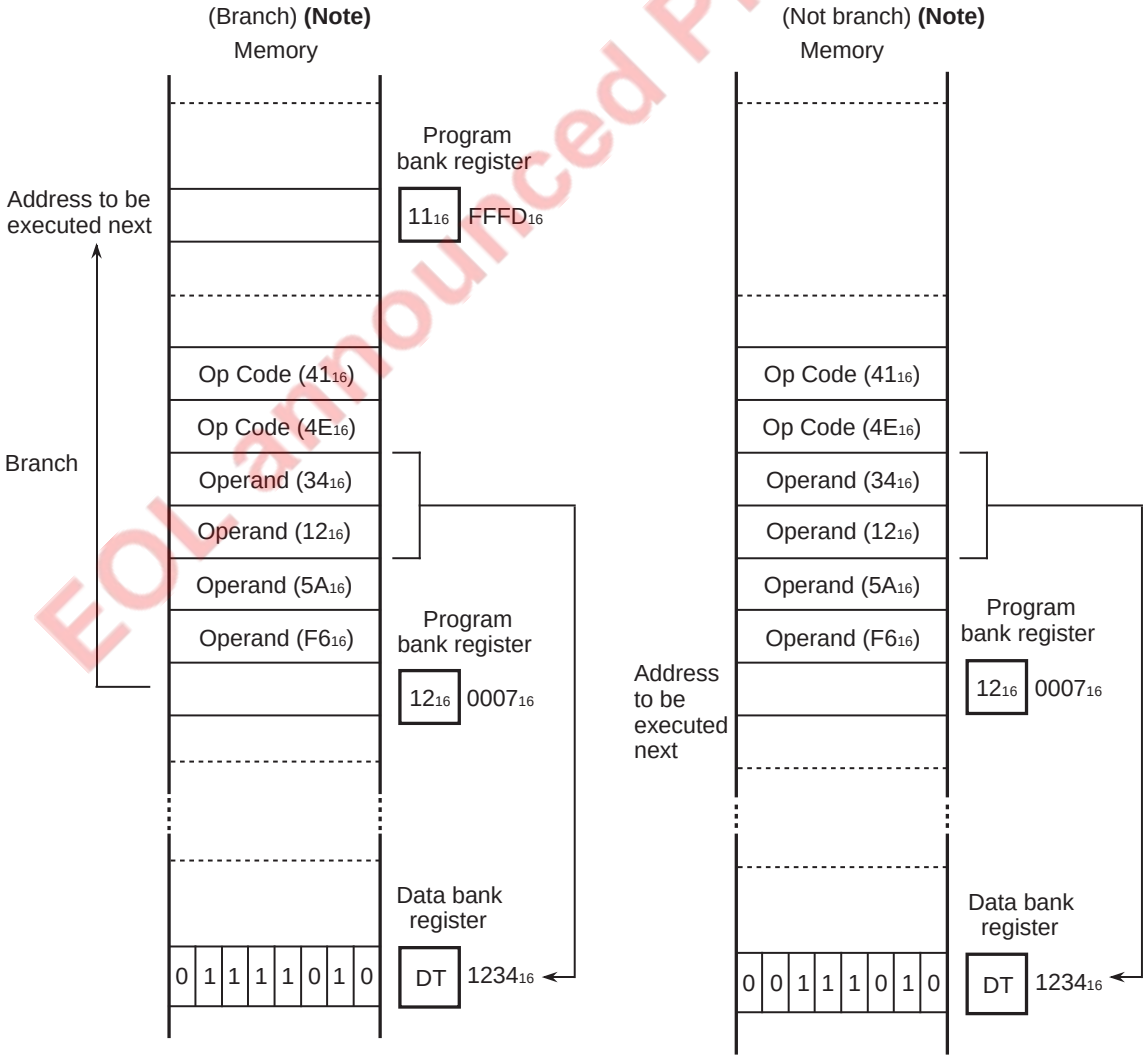
Mode : Absolute bit relative addressing mode

Function :

- BBC and BBS instructions
  Specifies the memory location by the instruction's third and fourth bytes and the contents of the data bank register; specifies the multiple bits' position in that memory by the bit pattern of the instruction's fifth and sixth bytes (when the m flag is "1," the fifth byte only). Then, when the specified bits all satisfy the branching conditions, branches to the address specified by the result of adding the instruction's seventh byte (or when the m flag is "1," the sixth byte) as a signed numerical value to the program counter's contents.
- BBCB and BBSB instructions
  Specifies the memory location by the instruction's second and third bytes and the contents of the data bank register; specifies the multiple bits' position in that memory by the bit pattern of the instruction's fourth byte. Then, when the specified bits all satisfy the branching conditions, branches to the address specified by the result of adding the instruction's fifth byte as a signed numerical value to the program counter's contents.

ex. : Mnemonic Machine code

BBS #5AH, 1234H, 0F6H 41_{16} $4E_{16}$ 34_{16} 12_{16} $5A_{16}$ $F6_{16}$
(m = "1")



Note: Whether to branch or not depends on the branching conditions.

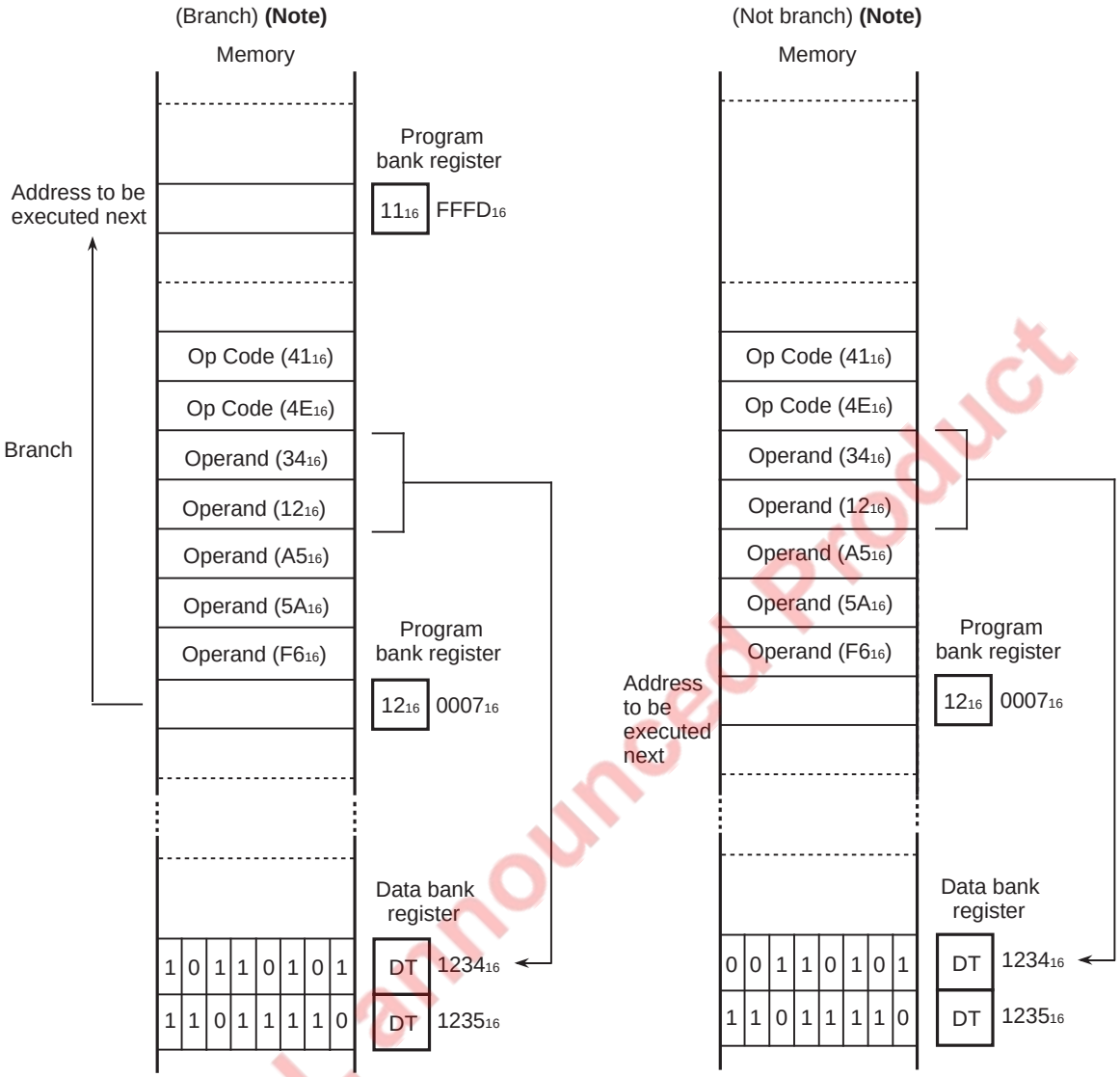
Absolute Bit Relative

ex. : Mnemonic

BBS #5AA5H, 1234H, 0F6H
(m = "0")

Machine code

41_{16} $4E_{16}$ 34_{16} 12_{16} $A5_{16}$ $5A_{16}$ $F6_{16}$



Note: Whether to branch or not depends on the branching conditions.

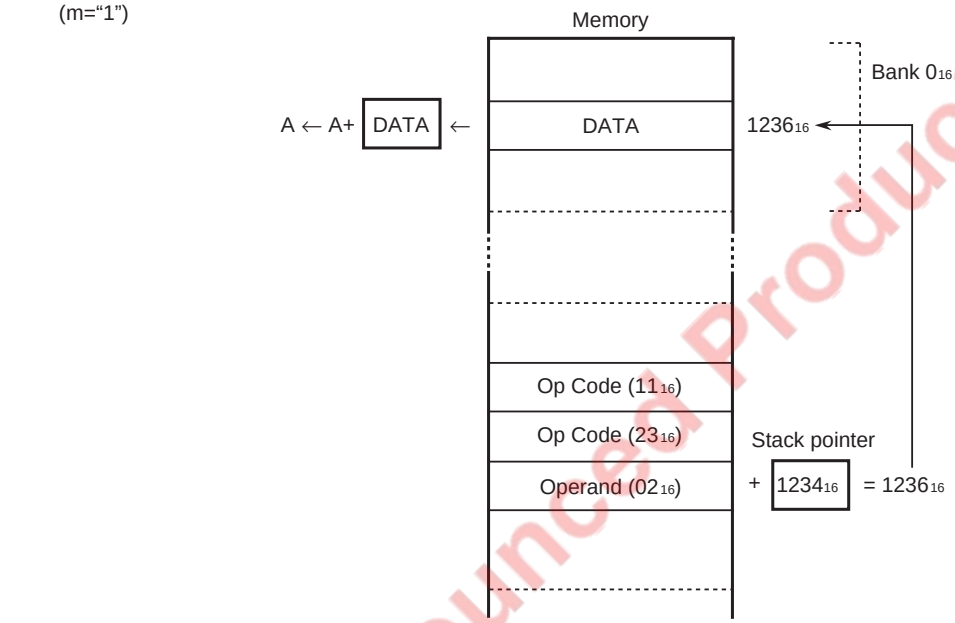
Stack Pointer Relative

Mode : Stack pointer relative addressing mode

Function : The contents of the memory location in bank 0_{16} are an actual data. This memory is specified by the result of adding the instruction's operand to the stack pointer's contents. When, however, the result of adding the instruction's operand to the stack pointer's contents exceeds the bank 0_{16} range, the memory location in bank 1_{16} is specified.

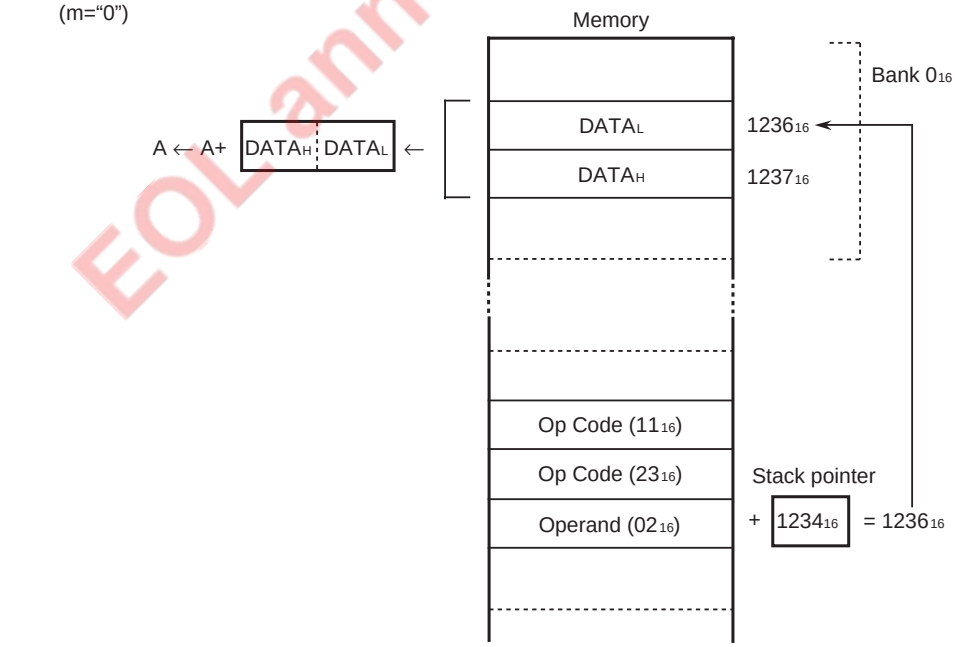
ex. : Mnemonic
ADD A, 02H, S
(m="1")

Machine code
11_{16} 23_{16} 02_{16}



ex. : Mnemonic
ADD A, 02H, S
(m="0")

Machine code
11_{16} 23_{16} 02_{16}



Stack Pointer Relative Indirect Indexed Y

Mode : Stack pointer relative indirect indexed Y addressing mode

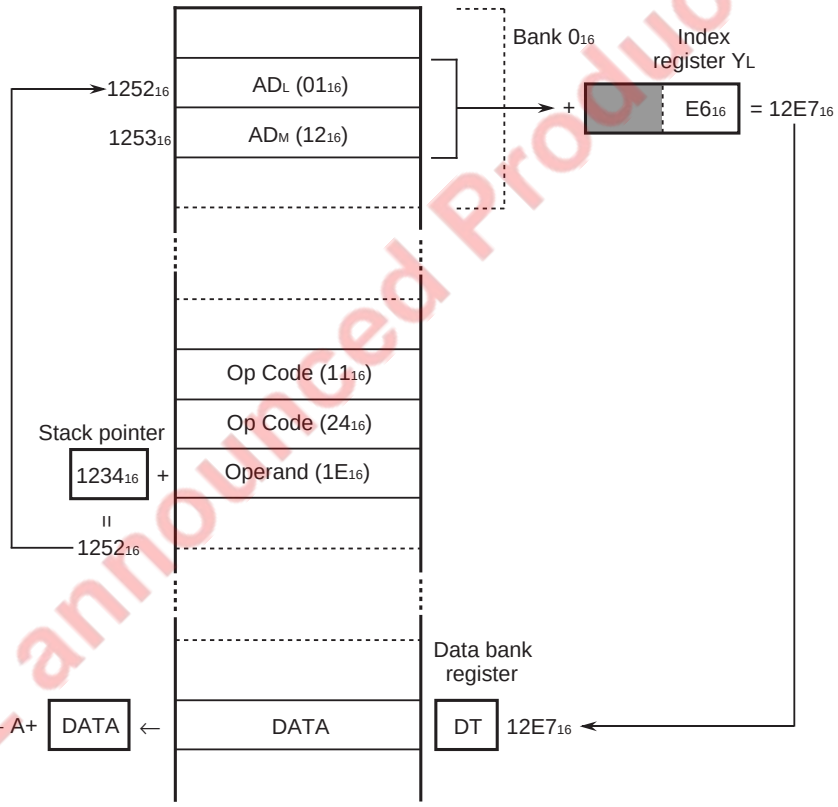
Function : Specifies a sequence of 2-byte memory by the result of adding the instruction's operand to the stack pointer's contents. The contents of the memory location specified by the above addition are added to the index register Y's contents. The result of second addition and the contents of data bank register DT indicate the memory location which contents an actual data. If, however, the result of adding the contents of that sequence of 2-byte memory to the index register Y's contents generates a carry, the value which is 1 larger than the contents of the data bank register DT indicates the bank.

ex. : Mnemonic

ADD A, (1EH, S), Y
(m = "1," x = "1")

Machine code

11_{16} 24_{16} $1E_{16}$



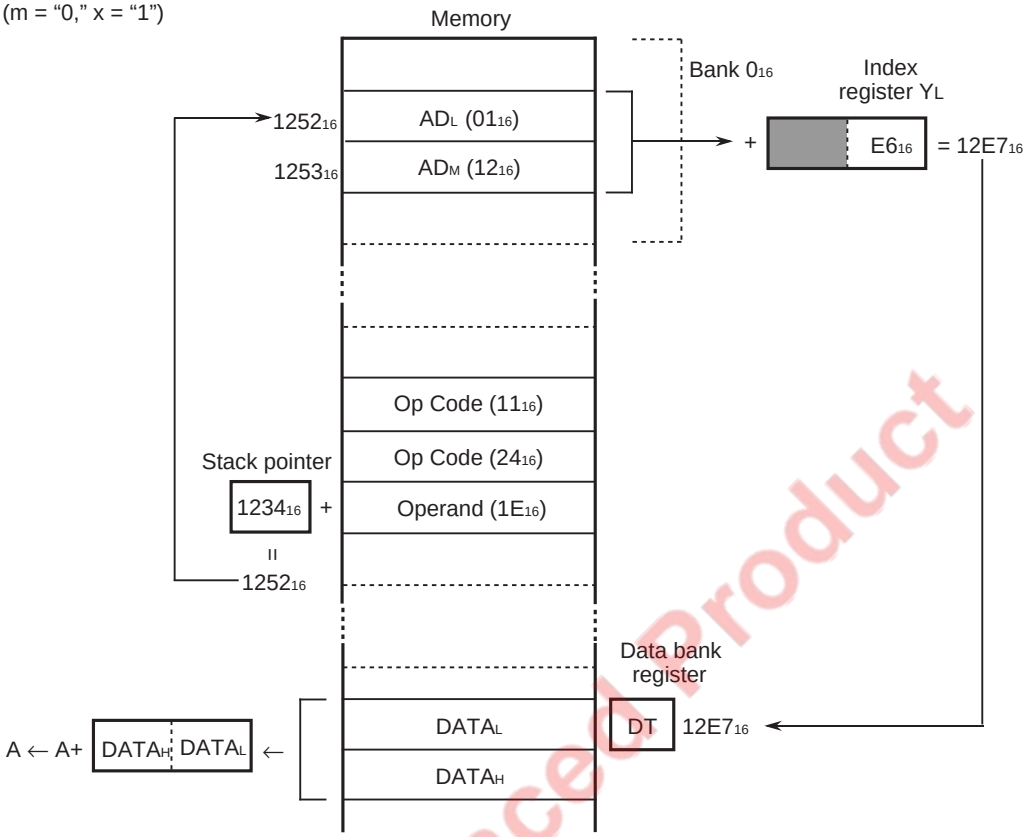
Stack Pointer Relative Indirect Indexed Y

ex. : Mnemonic

ADD A, (1EH, S), Y
(m = "0," x = "1")

Machine code

11_{16} 24_{16} $1E_{16}$

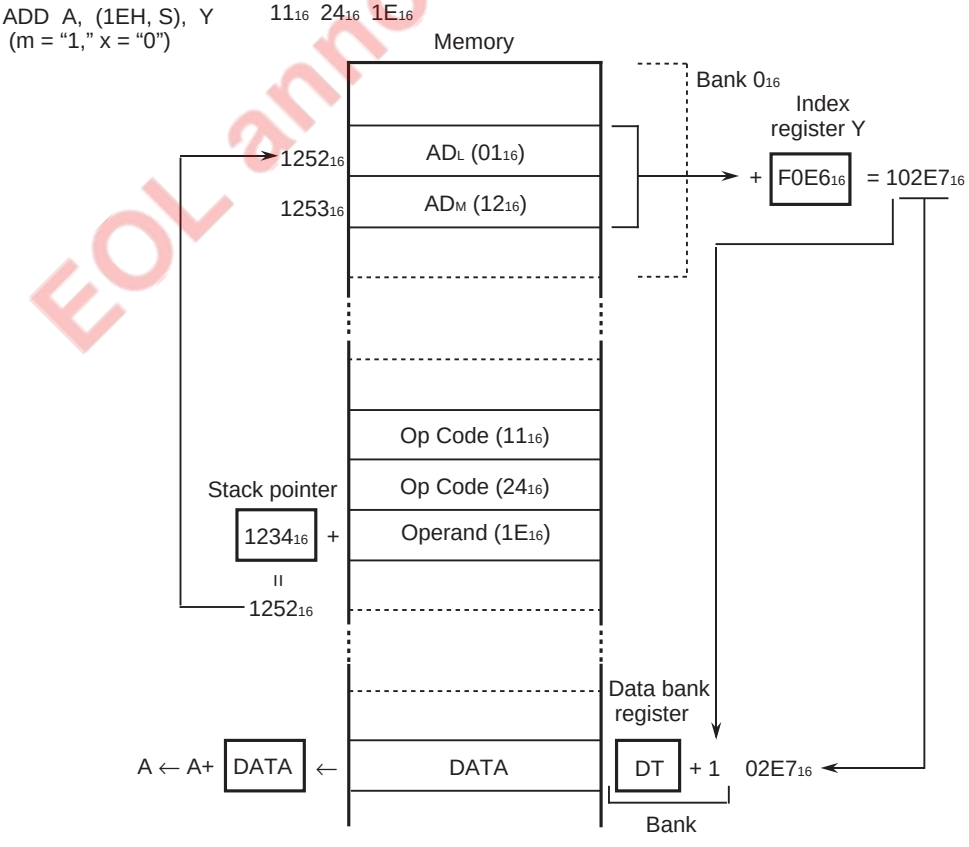


ex. : Mnemonic

ADD A, (1EH, S), Y
(m = "1," x = "0")

Machine code

11_{16} 24_{16} $1E_{16}$



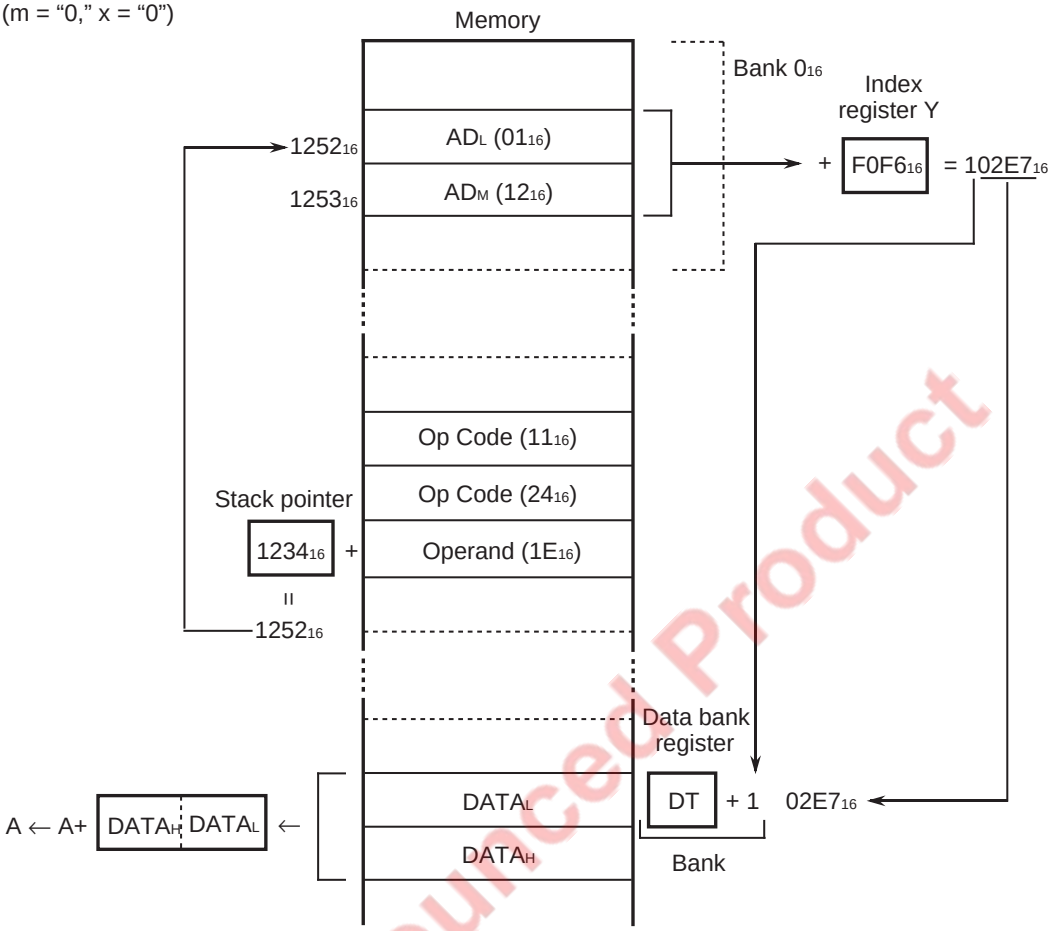
Stack Pointer Relative Indirect Indexed Y

ex.: Mnemonic

ADD A, (1EH, S), Y
(m = "0," x = "0")

Machine code

11_{16} 24_{16} $1E_{16}$

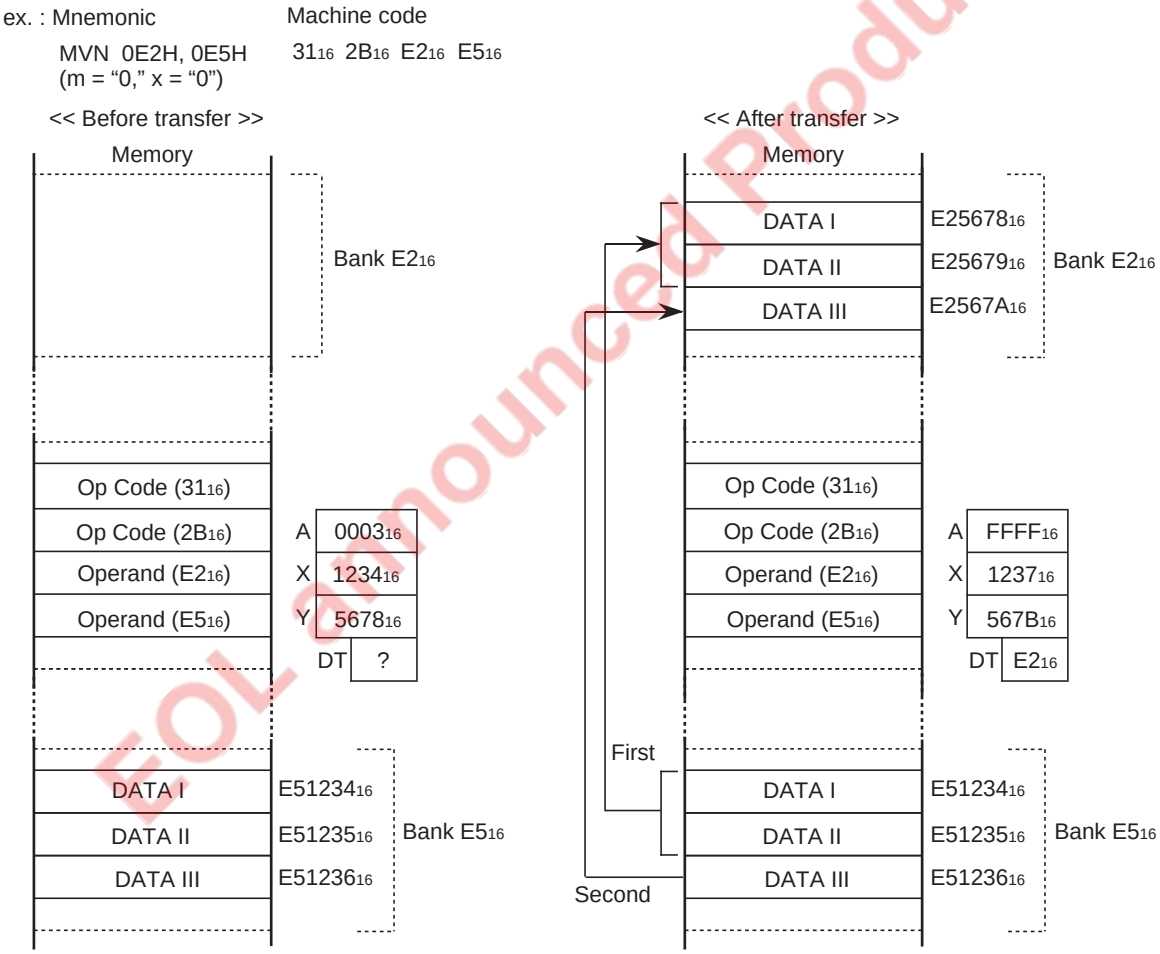


Block Transfer

Mode : Block transfer addressing mode

Function : Specifies the transfer destination data bank by the instruction's third byte, and specifies the transfer destination address within the data bank by the index register Y's contents. Specifies the transfer source data bank by the instruction's fourth byte, and specifies the address of transfer data within the data bank by the index register X's contents. The accumulator A's contents are the number of bytes to be transferred. At termination of transfer, the data bank register's contents specify the transfer destination data bank.

- MVN instruction
  The MVN instruction is used for transfer toward lower addresses. In this case, the contents of index registers X and Y are incremented each time data is transferred.
- MVP instruction
  The MVP instruction is used for transfer toward higher addresses. In this case, the contents of index registers X and Y are decremented each time data is transferred. The transfer data can cross over the bank boundary.

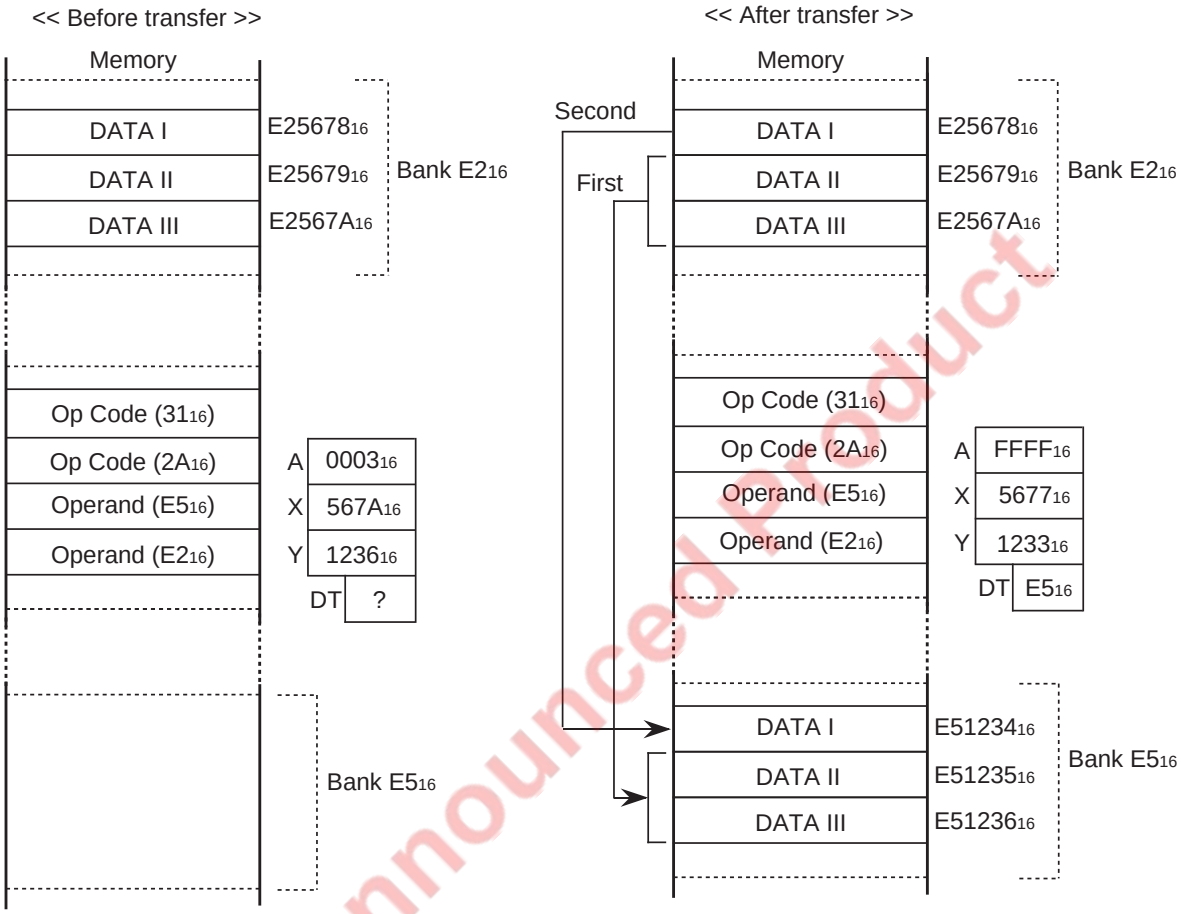


ex. : Mnemonic

MVP 0E5H, 0E2H
(m = "0," x = "0")

Machine code

31_{16} $2A_{16}$ $E5_{16}$ $E2_{16}$

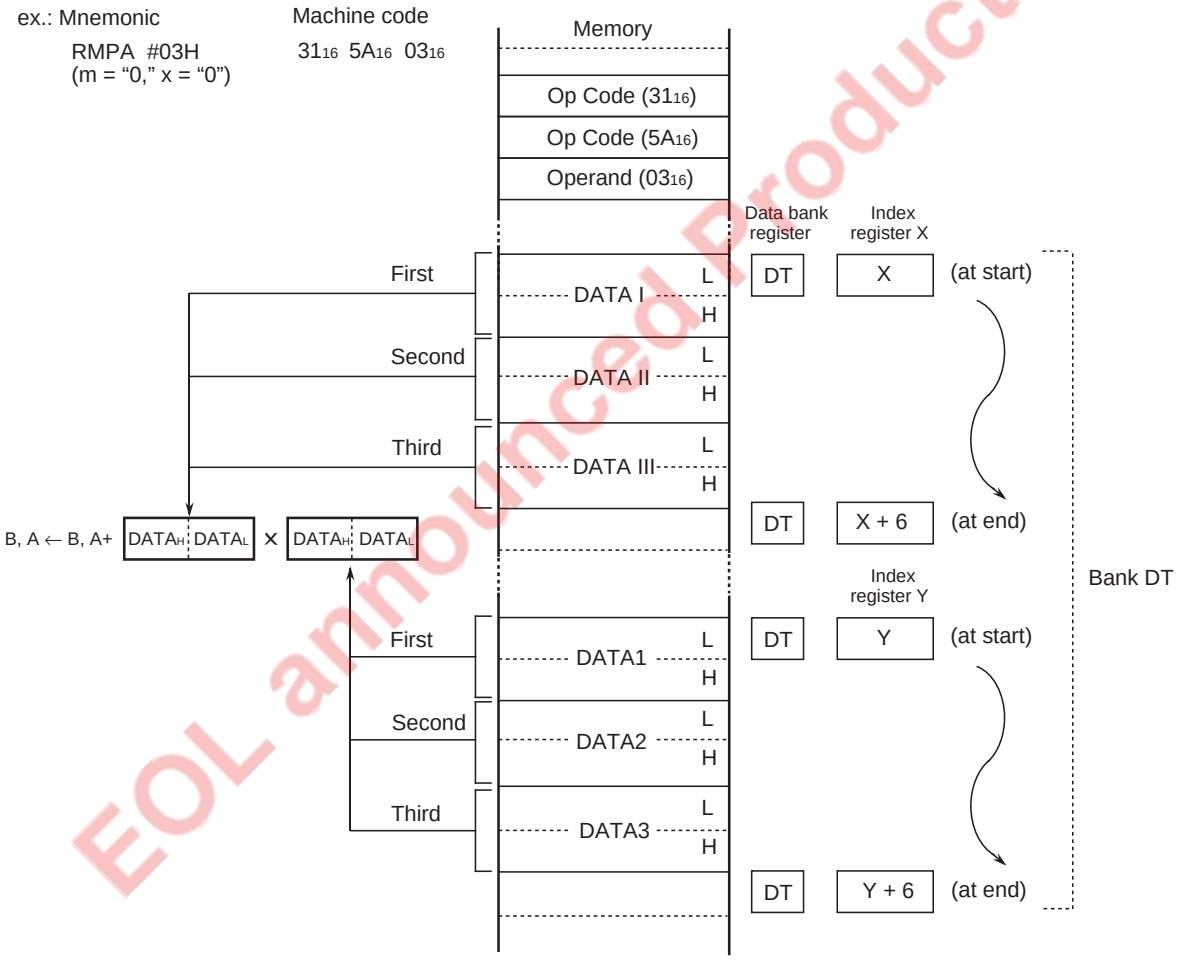


Note : For block transfer instructions, the number of bytes to be transferred and the range can be specified as transfer source/destination addresses change with the state of the m and x flags. However, the transfer unit is unaffected. The transfer unit is "word" (16 bits). However, only 1 byte is transferred when transferring the last byte at odd-byte transfer.

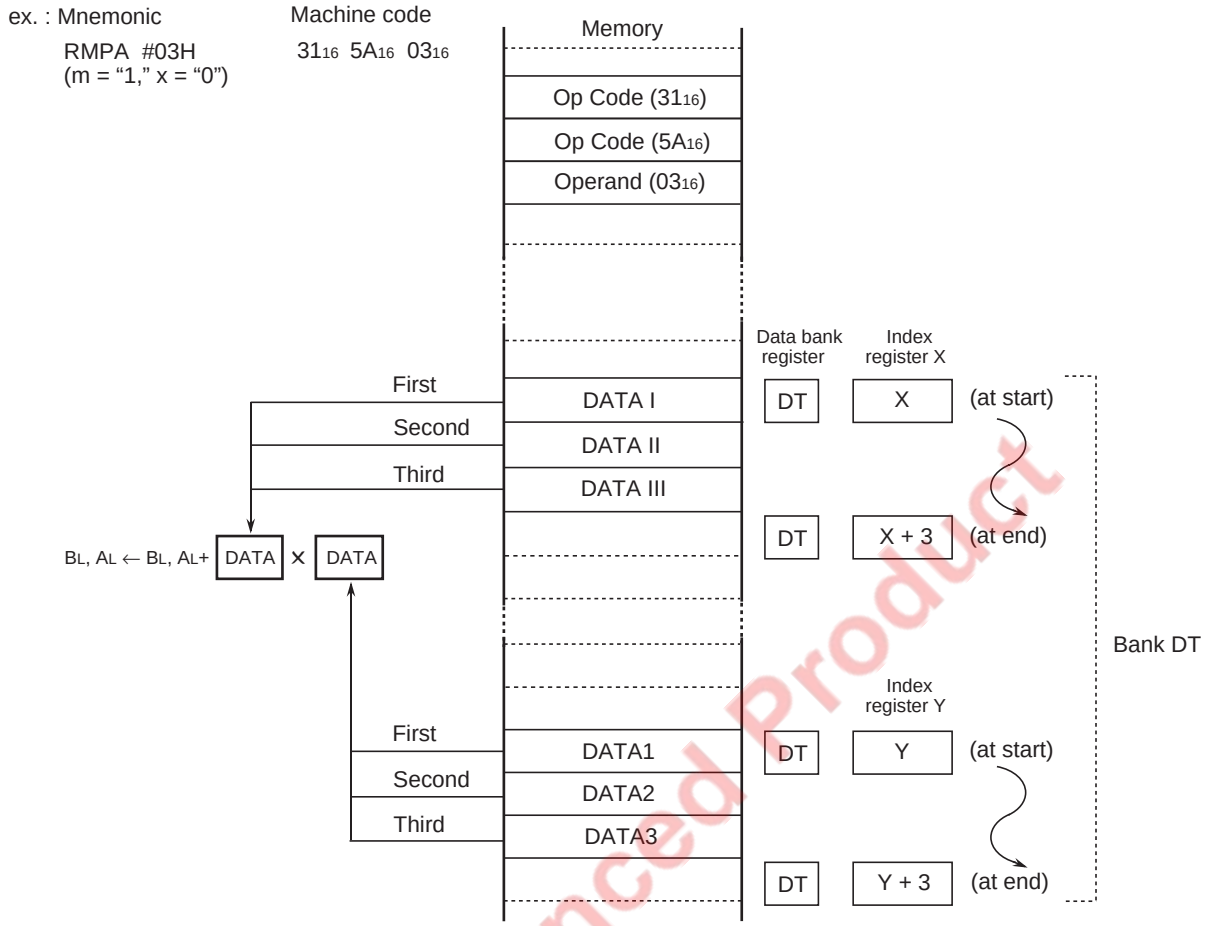
Multiplied accumulation

Mode : Multiplied accumulation addressing mode

Function : The following is a multiplicand and a multiplier: the contents of the memory location specified by the contents of index registers X and Y, and the data bank register's contents. The instruction's third byte is the repeat number of arithmetic operation. The contents of index registers X and Y are incremented each time the addition of the contents of accumulators B and A to the multiplication result finishes. Accordingly, the contents of index registers X and Y specify the next address where the multiplicand and the multiplier are read at last.
Allocate a multiplicand and a multiplier within the same bank and do not cross them over the bank boundary.
Set index register length flag x to "0" before executing this instruction.
This addressing mode is used by an RMPA instruction.



Multiplied accumulation



CHAPTER 3

HOW TO USE 7900 SERIES INSTRUCTIONS

3.1 Memory access

Memory access modes are typically classified into the following 3 categories:

- Direct addressing
- Absolute addressing and Absolute long addressing
- Indirect addressing and Indirect long addressing

Their features are described below.

3.1.1 Direct addressing

- Each instruction has a length of 2 or 3 bytes.
- Reduced number of consumed instruction execution cycles.
- A block (within bank 0: addresses 000000_{16}–$00FFFF_{16}$) of which base address is specified by DPRn is addressable.
  (i) Direct page register select bit is "0":
  Block size = 256 bytes
  (ii) Direct page register select bit is "1":
  Block size = 64 bytes
  When a sum of DPRn's contents and an offset value exceeds the bank boundary, however, access over the boundary is enabled.

3.1.2 Absolute addressing and Absolute long addressing

(1) Absolute addressing

- Each instruction has a length of 3 or 4 bytes.
- A 64-Kbyte space (a bank within addresses 000000_{16}–$FFFFFF_{16}$) is addressable, where the high-order 8 bits of 24-bit address are specified by DT. For the JMP and JSR instructions, however, these high-order 8 bits are specified by PG.

(2) Absolute long addressing

- Each instruction has a length of 4 or 5 bytes.
- Addresses 000000_{16}–$FFFFFF_{16}$ are addressable. All of 24 bits of the address are directly specified.

3.1.3 Indirect addressing and Indirect long addressing

(1) Direct indirect addressing

- Each instruction has a length of 2 or 3 bytes.
- 16-bit pointer data is placed in the space specified by DPRn, and the specified memory is accessed.
- A 64-KB space (a bank within addresses 000000_{16}–$FFFFFF_{16}$) is addressable, where the high-order 8 bits of 24-bit address are specified by DT.

(2) Direct indirect long addressing

- Each instruction has a length of 2 or 3 bytes.
- 24-bit pointer data is placed in the space specified by DPRn, and the specified memory is accessed.
- An address within the 16-Mbyte space (addresses 000000_{16}–$FFFFFF_{16}$) is addressable.

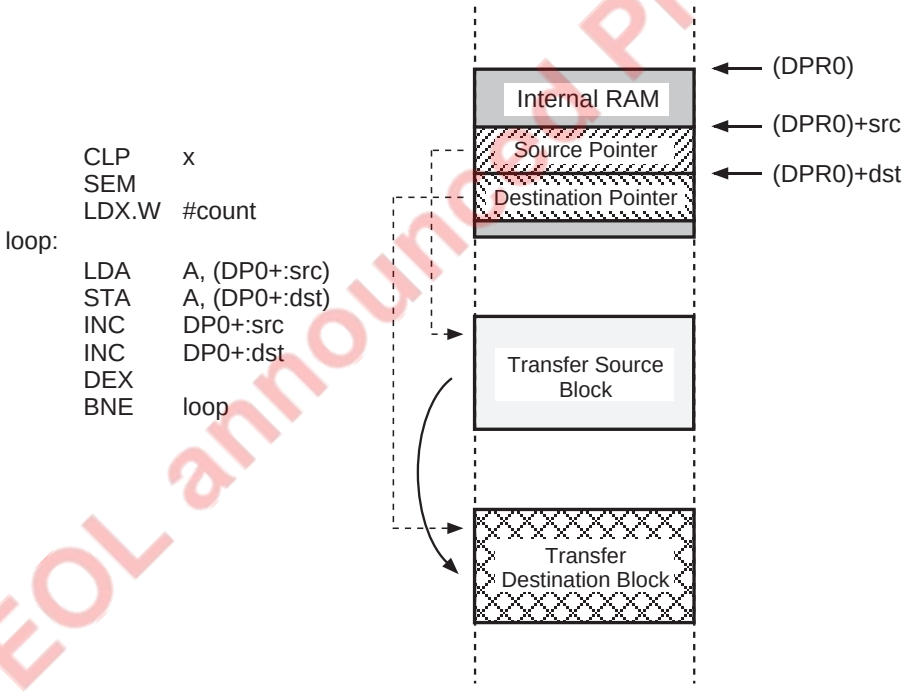
(3) Absolute indirect addressing

- This addressing mode can be used only for the indirect branch and indirect subroutine call instructions.
- Each instruction has a length of 3 or 4 bytes.
- 16-bit pointer data is placed in the space specified by PG, and the specified memory is accessed.
- A 64-KB space (a bank within addresses 000000_{16}–$FFFFFF_{16}$) is addressable, where the high-order 8 bits of 24-bit address are specified by PG.

(4) Absolute indirect long addressing

- This addressing mode can be used only for the indirect branch instruction.
- Each instruction has a length of 3 or 4 bytes.
- 24-bit pointer data is placed in the space specified by PG, and the specified memory is accessed.
- Any address of the 16-Mbyte space (addresses 000000_{16}–$FFFFFF_{16}$) is addressable.

Figure 3.1.1 shows a usage example of indirect addressing mode.
Here, the data of the pointers pointing to memory areas are processed in the program, and the results are referenced as effective addresses.



❋ For description format of mnemonics, refer to the tool's specifications.

Fig. 3.1.1 Usage example of indirect addressing mode: block transfer

The 7900 Series also provides many other useful addressing modes. For details, refer to section **"2.3 Addressing modes."**

3.2 Direct page registers (DPR0–DPR3)

The 7900 Series provides more enhanced direct addressing modes than those of the conventional 7700 Family. These powerful addressing modes greatly improve programming efficiency, especially in a range of addresses 000000_{16}–$00FFFF_{16}$.

In the 7900 Series, just after a reset, only DPR0 can be used. When the direct page register select bit of the processor mode register 1 is set to "1," however, direct page registers DPR0–DPR3 can be used. Figure 3.2.1 shows an usage example of DPR0–DPR3.

In the conventional 7700 Family, since only one direct page register can be used, it is required to frequently change the contents of the direct page register for efficient memory access using direct page addressing mode. On the contrary, the 7900 Series does not need such a procedure as in the conventional 7700 Family because it can assign a direct page register to each base address of each block.

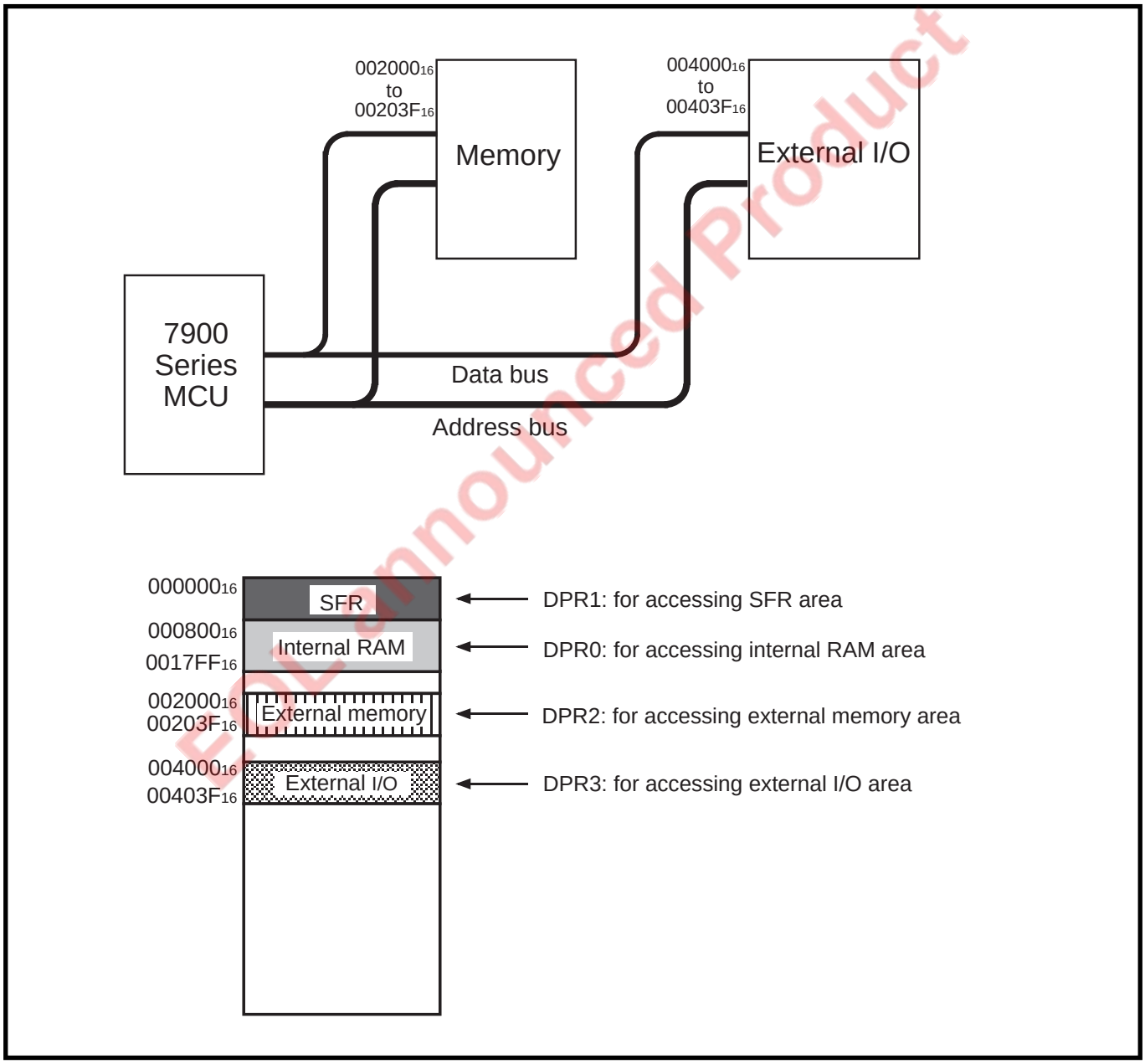
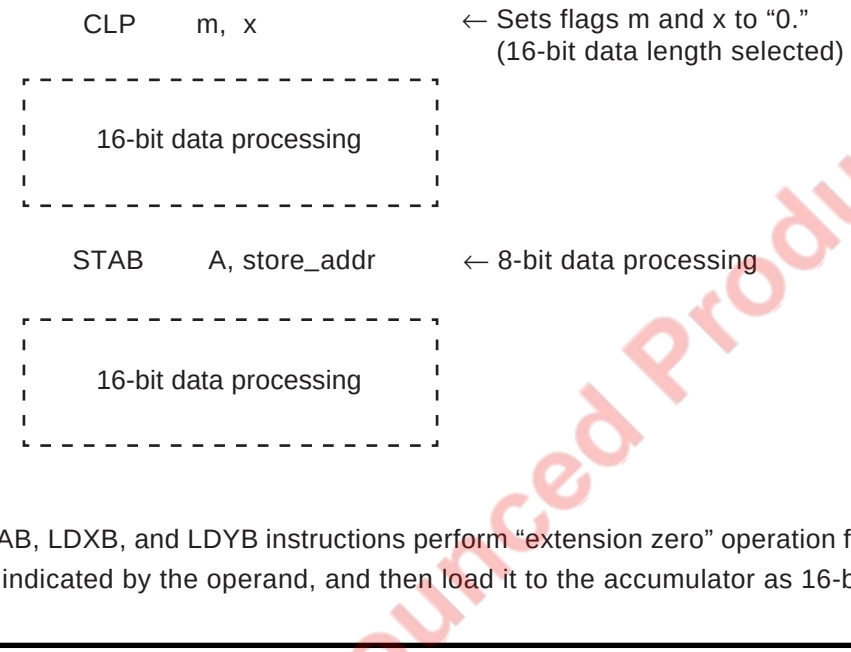


Fig. 3.2.1 Usage example of DPR0–DPR3

3.3 8- and 16-bit data processing

In the conventional 7700 Family, the same machine code is assigned to an 8- and its corresponding 16-bit instruction in order to reduce program size, so that it is necessary to specify whether 8- or 16-bit data is processed, by using flags m and x. The 7900 Series incorporates new instructions with the conventional instructions. These new instructions enable 8-bit operation independent of flags m and x. By using these new instructions, 8-bit data can be processed while flags are set for 16-bit data length, preventing an overhead generated by setting flags. Figure 3.3.1 shows an 8-bit operation example.



Note: LDAB, LDXB, and LDYB instructions perform "extension zero" operation for 8-bit data which is indicated by the operand, and then load it to the accumulator as 16-bit data.

Fig. 3.3.1 8-bit operation example

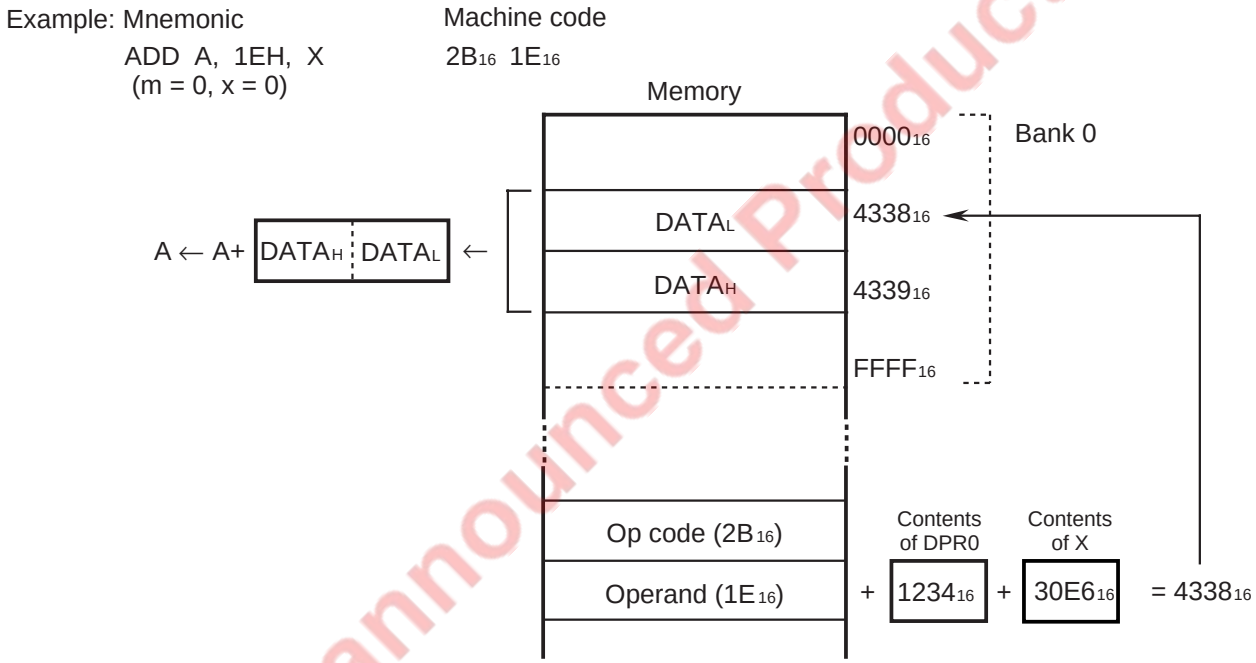
When executing the instructions that require the data length setting by flags m and x, the number of bytes or execution cycles is affected by this setting. For details, refer to section **"4.2 Description of each instruction"** or **"Appendix 1. 7900 Series machine instructions."**

3.4 Index registers X and Y

The contents of index register X or Y facilitate to specify an effective address. For example, the direct indexed X addressing mode is described below. Refer to section **"2.3 Addressing modes"** for details.

<Example> Direct indexed X addressing mode

A sum of the instruction's operand, the contents of a direct page register, and the contents of index register X indicates a memory location in bank 0. The contents in this memory location are data to be processed. However, when the above sum exceeds the boundary of bank 0 or bank 1, a memory location in bank 1 or bank 2 is specified, respectively.



3.5 Branch instructions

The branch instructions are classified into the following 6 categories:
(1) Relative branch
(2) Absolute branches (absolute and absolute long)
(3) Indirect branches (absolute indirect and absolute indirect long)
(4) Relative subroutine call
(5) Absolute subroutine calls (absolute and absolute long)
(6) Indirect subroutine call (absolute Indexed X indirect)

Relative branch and relative subroutine call instructions have the following features:
- Each instruction has a length of 2 or 3 bytes.
- Program area can be reallocated dynamically during program execution.
- Addresses to which the program can branch are limited within a specified range. Refer to section **"4.2 Description of each instruction"** for details.

Examples:
(i) BRA instruction ··· Within a range of –128 to +127 referenced to PC just after instruction execution
(ii) BRAL instruction ··· Within a range of –32768 to +32767 referenced to PC just after instruction execution
(iii) BSR instruction ··· Within a range of –1024 to +1023 referenced to PC just after instruction execution

On the other hand, absolute branch, absolute subroutine call, indirect branch and indirect subroutine call instructions have the following features:
- Any address within the 16-Mbyte space can be directly specified as a branch destination address (absolute long).
- Any address limited within the 64-Kbyte space (a bank), containing PC being used, also can be specified as a branch destination address. In this case, byte length of an instruction and the number of instruction execution cycles can be reduced. Refer to section **"4.2 Description of each instruction"** for details.

Examples:
(i) JMP instruction ··· Branches to a 64-Kbyte space specified by PG in which the last byte of an instruction is located.
(ii) JMPL instruction ··· Branches to a specified address within the 16-Mbyte space.
(iii) JSR instruction ··· Branches to a 64-Kbyte space specified by PG in which the last byte of an instruction is located. Returns from the branch destination address by the RTS instruction.
(iv) JSRL instruction ··· Branches to a specified address within the 16-Mbyte space. Returns from the branch destination address by the RTL instruction.

Figure 3.5.1 shows the branch examples by JMP/JMPL and JSR/JSRL instructions.

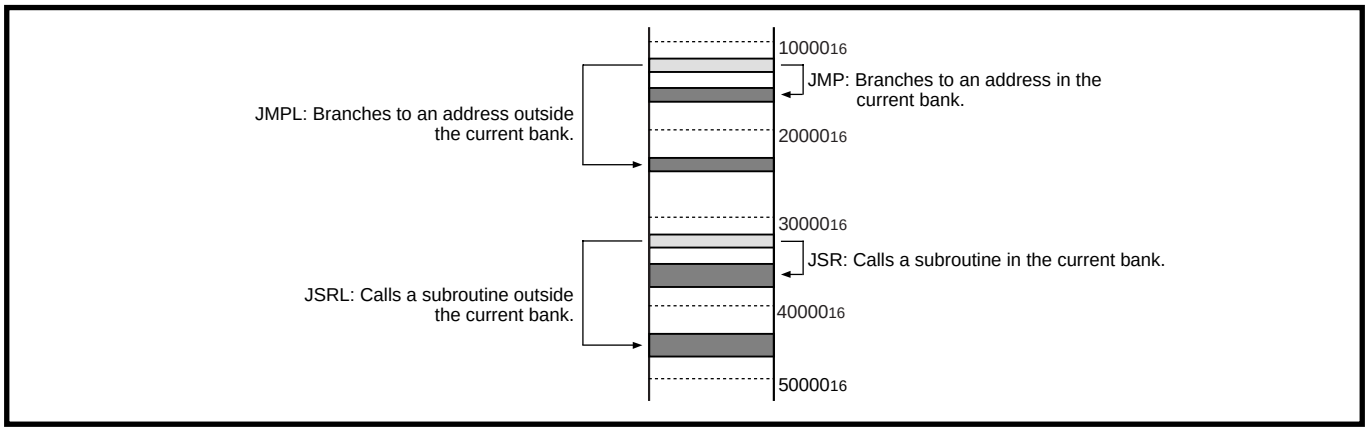


Fig. 3.5.1 Branch examples by JMP/JMPL and JSR/JSRL instructions

CHAPTER 4
INSTRUCTIONS

4.1 Instruction set

The 7900 Series CPU uses the instruction set with 203 instructions.
Instructions marked by * are the new instructions that have been added to the 7751 Series instruction set. The remarks column shows that a conventional 7700 Family's instruction is included in the corresponding new instruction.

Category	Instruction	Description	Remarks
Load	LDA	Acc ←M	
	* LDAB	Acc ←M8 (Extended with "0"s.)	
	* LDAD	E ←M32	
	* LDD n	DPRn←IMM16 (n = 0 to 3. Multiple operations can be specified.)	
	LDT	DT ←IMM8	
	LDX	X ←M	
	* LDXB	X ←IMM8 (Extended with "0"s.)	
	LDY	Y ←M	
	* LDYB	Y ←IMM8 (Extended with "0"s.)	
Store	STA	M ←Acc	
	* STAB	M8 ←Acc_L	
	* STAD	M32 ←E	
	STX	M ←X	
	STY	M ←Y	
Transfer between registers	* TAD n	DPRn←A (n = 0 to 3)	including TAD instruction
	TAS	S ←A	
	TAX	X ←A	
	TAY	Y ←A	
	* TBD n	DPRn←B (n = 0 to 3)	including TBD instruction
	TBS	S ←B	
	TBX	X ←B	
	TBY	Y ←B	
	* TDA n	A ←DPRn (n = 0 to 3)	including TDA instruction
	* TDB n	B ←DPRn (n = 0 to 3)	including TDB instruction
	* TDS	S ←DPR0	
	TSA	A ←S	
	TSB	B ←S	
	* TSD	DPR0←S	
	TSX	X ←S	
	TXA	A ←X	
	TXB	B ←X	
	TXS	S ←X	
	TXY	Y ←X	
	TYA	A ←Y	
	TYB	B ←Y	
	TYX	X ←Y	
	XAB	A $\rightleftarrows$B	

Category	Instruction	Description	Remarks
Transfer between memories	* MOVM	M ←M	including LDM instruction
	* MOVMB	M8 ←M8	
	* MOVR	M(dest n) ←M(source n) (Multiple operations can be specified.) (n = 0 to 15)	
	* MOVRB	M8(dest n) ←M8(source n) (Multiple operations can be specified.) (n = 0 to 15)	
Block transfer	MVN	M (n to n + i – 1) ←M (m to m + i – 1) (i:transfer byte number)	
	MVP	M (n – i + 1 to n) ←M (m – i + 1 to m) (i:transfer byte number)	
Stack operation	PEA	Stack←IMM16	
	PEI	Stack←M16 (DPRn + dd) (n = 0 to 3)	
	PER	Stack←PC + IMM16	
	PHA	Stack←A	
	PHB	Stack←B	
	PHD	Stack←DPR0	
	* PHD n	Stack←DPRn (n = 0 to 3. Multiple operations can be specified.)	
	PHG	Stack←PG	
	PHP	Stack←PS	
	PHT	Stack←DT	
	PHX	Stack←X	
	PHY	Stack←Y	
	PLA	A ←Stack	
	PLB	B ←Stack	
	PLD	DPR0←Stack	
	* PLD n	DPRn←Stack (n = 0 to 3. Multiple operations can be specified.)	
	PLP	PS ←Stack	
	PLT	DT ←Stack	
	PLX	X ←Stack	
	PLY	Y ←Stack	
	PSH	Stack ←Any specified register among A, B, X, Y, DPR0, DT, PG, and PS. (Multiple operations can be specified) M (S to S – i + 1) ←A, B, X, Y, DPR0, DT, PG, PS S ←S – i (i : Number of bytes corresponding to the registers saved to the stack.)	
	PUL	Any specified register among A, B, X, Y, DPR0, DT, and PS. ←Stack (Multiple operations can be specified) A, B, X, Y, DPR0, DT, PS ←M (S + 1 to S + i) S ←S + i (i : Number of bytes corresponding to the registers restored from the stack.)	

INSTRUCTIONS

4.1 Instruction set

Category	Instruction	Description	Remarks
Stack operation & Load	* PHLD n	stack ←DPRn, DPRn ←IMM16 (n = 0 to 3. Multiple operations can be specified)	
Clearance	* CLR	Acc ←0	
	* CLRB	Acc_L ←0	
	* CLRM	M ←0	
	* CLRMB	M8 ←0	
	* CLRX	X ←0	
	* CLRY	Y ←0	
Addition	ADC	Acc ←Acc + M + C	
	* ADCB	Acc_L ←Acc_L + IMM8 + C	
	* ADCD	E ←E + M32 + C	
	* ADD	Acc ←Acc + M	
	* ADDB	Acc_L ←Acc_L + IMM8	
	* ADDD	E ←E + M32	
	* ADDM	M ←M + IMM	
	* ADDMB	M8 ←M8 + IMM8	
	* ADDMD	M32 ←M32 + IMM32	
	* ADDS	S ←S + IMM8	
	* ADDX	X ←X + IMM (IMM = 0 to 31)	
	* ADDY	Y ←Y + IMM (IMM = 0 to 31)	
Increment	INC	Acc ←Acc + 1 or M ←M + 1	
	INX	X ←X + 1	
	INY	Y ←Y + 1	
Subtraction	SBC	Acc ←Acc – M – $\overline{C}$	
	* SBCB	Acc_L ←Acc_L – IMM8 – $\overline{C}$	
	* SBCD	E ←E – M32 – $\overline{C}$	
	* SUB	Acc ←Acc – M	
	* SUBB	Acc_L ←Acc_L – IMM8	
	* SUBD	E ←E – M32	
	* SUBM	M ←M – IMM	
	* SUBMB	M8 ←M8 – IMM8	
	* SUBMD	M32 ←M32 – IMM32	
	* SUBS	S ←S – IMM8	
	* SUBX	X ←X – IMM (IMM = 0 to 31)	
	* SUBY	Y ←Y – IMM (IMM = 0 to 31)	
Decrement	DEC	Acc ←Acc – 1 or M ←M – 1	
	DEX	X ←X – 1	
	DEY	Y ←Y – 1	
Multiplication	MPY	(B, A) ←A (Multiplicand) × M (Multiplier), Unsigned	
	MPYS	(B, A) ←A (Multiplicand) × M (Multiplier), Signed	
Division	DIV	A (Quoitent), B (remainder)←(B, A) ÷ M, Unsigned	
	DIVS	A (Quoitent), B (remainder)←(B, A) ÷ M, Signed	
Multiplied accumulation	RMPA	(B, A) ←(B, A) + M (DT:X) × M (DT:Y) (repeating 0 to 255 times)	

Category	Instruction	Description	Remarks
Logical OR	ORA	Acc ←Acc∨M	
	* ORAB	Acc_L ←Acc_L∨IMM8	
	* ORAM	M ←M∨IMM	Including SEB instruction
	* ORAMB	M8 ←M8∨IMM8	
	* ORAMD	M32 ←M32∨IMM32	
Logical AND	AND	Acc ←Acc∧M	
	* ANDB	Acc_L ←Acc_L∧IMM8	
	* ANDM	M ←M∧IMM	Including CLB instruction
	* ANDMB	M8 ←M8∧IMM8	
	* ANDMD	M32 ←M32∧IMM32	
Logical exclusive OR	EOR	Acc ←Acc∀M	
	* EORB	Acc_L ←Acc_L∀IMM8	
	* EORM	M ←M∀IMM	
	* EORMB	M8 ←M8∀IMM8	
	* EORMD	M32 ←M32∀IMM32	
Comparison	CMP	Acc – M	
	* CMPB	Acc_L – IMM8	
	* CMPD	E – IMM32	
	* CMPM	M – IMM	
	* CMPMB	M8 – IMM8	
	* CMPMD	M32 – IMM32	
	CPX	X – M	
	CPY	Y – M	
Arithmetic shift left	ASL	Shifts the contents of Acc or M to the left by 1 bit.	
	* ASL #n	Shifts the contents of A to the left by n bits (n = 0 to 15).	
	* ASLD #n	Shifts the contents of E to the left by n bits (n = 0 to 31).	
Arithmetic shift right	ASR	Shifts the contents of Acc or M holding a sign to the right by 1 bit.	
	* ASR #n	Shifts the contents of A holding a sign to the right by n bits (n = 0 to 15).	
	* ASRD #n	Shifts the contents of E holding a sign to the right by n bits (n = 0 to 31).	
Logical shift right	LSR	Shifts the contents of Acc or M to the right by 1 bit.	
	* LSR #n	Shifts the contents of A to the right by n bits (n = 0 to 15).	
	* LSRD #n	Shifts the contents of E to the right by n bits (n = 0 to 31).	

INSTRUCTIONS

4.1 Instruction set

Category	Instruction	Description	Remarks
Rotation to left	RLA	Rotates the contents of A to the left by n bits. (When m = 0:n = 0 to 65535, when m = 1:n = 0 to 255)	
	ROL	Links the contents of Acc or M with C, and rotates the result to the left by 1 bit.	
	* ROL #n	Links the contents of A with C, and rotates the result to the left by n bits (n = 0 to 15).	
	* ROLD #n	Links the contents of E with C, and rotates the result to the left by n bits (n = 0 to 31).	
Rotation to right	ROR	Links the contents of Acc or M with C, and rotates the result to the right by 1 bit.	
	* ROR #n	Links the contents of A with C, and rotates the result to the right by n bits (n = 0 to 15).	
	* RORD #n	Links the contents of E with C, and rotates the result to the right by n bits (n = 0 to 31).	
Extension Sign	EXTS	Acc ←Acc_L (Extended with a sign.)	
	* EXTSD	E ←E_L (= A) (Extended with a sign.)	
Extension Zero	EXTZ	Acc ←Acc_L (Extended with “0”s.)	
	* EXTZD	E ←E_L (= A) (Extended with “0”s.)	
Sign invertion	* NEG	Acc ← –Acc	
	* NEGD	E ← –E	
Absolute value	* ABS	Acc ←| Acc |	
	* ABSD	E ←| E |	
Flag manipulation	CLC	C ←0	
	CLI	I ←0	
	CLM	m ←0	
	CLP	PS_L(bit n)←0 (n = 0 to 7. Multiple operations can be specified.)	
	CLV	V ←0	
	SEC	C ←1	
	SEI	I ←1	
	SEM	m ←1	
	SEP	PS_L(bit n)←1 (n = 0 to 7. Multiple operations can be specified.)	
Conditional branch	BRA/BRAL	PC ←PC + cnt + REL (cnt : bytes number of BRA/BRAL instruction)	
	JMP	PC ←Destination address PC ←mmll	
	JMPL	PG, PC ←Destination address PC ←mmll PG ←hh	

Category	Instruction	Description	Remarks
Subroutine call	* BSR	Stack ←PC PC ←PC + 2 + REL	
	JSR	Stack ←PC PC ←Destination address PC ←PC + 3 M(S, S – 1) ←PC S ←S – 2 PC ←mmll	
	JSRL	Stack ←PG, PC PG, PC ←Destination address PC ←PC + 4 M(S, S – 2) ←PG, PC S ←S – 3 PC ←mmll PG ←hh	
Conditional branch	BBC	Branches relatively when the specified bits of M are all “0.”	
	* BBCB	Branches relatively when the specified bits of M8 are all “0.”	
	BBS	Branches relatively when the specified bits of M are all “1.”	
	* BBSB	Branches relatively when the specified bits of M8 are all “1.”	
	BCC	Branches relatively when C = 0.	
	BCS	Branches relatively when C = 1.	
	BEQ	Branches relatively when Z = 1.	
	* BGE	Branches relatively when N∀V = 0.	
	* BGT	Branches relatively when Z = 0 and N∀V = 0.	
	* BGTU	Branches relatively when C = 1 and Z = 0.	
	* BLE	Branches relatively when Z = 1 or N∀V = 1.	
	* BLEU	Branches relatively when C = 0 and Z = 1.	
	* BLT	Branches relatively when N∀V = 1.	
	BMI	Branches relatively when N = 1.	
	BNE	Branches relatively when Z = 0.	
	BPL	Branches relatively when N = 0.	
	* BSC	Branches relatively when the specified one bit of A or M is “0.”	
	* BSS	Branches relatively when the specified one bit of A or M is “1.”	
	BVC	Branches relatively when V = 0.	
	BVS	Branches relatively when V = 1.	
Compare & Conditional branch	* CBEQ	Branches relatively when Acc = IMM or M = IMM.	
	* CBEQB	Branches relatively when Acc_L = IMM8 or M8 = IMM8.	
	* CBNE	Branches relatively when Acc ≠ IMM or M ≠ IMM.	
	* CBNEB	Branches relatively when Acc_L ≠ IMM8 or M8 ≠ IMM8.	

INSTRUCTIONS

4.1 Instruction set

Category	Instruction	Description	Remarks
Decrement & Conditional branch	* DEBNE	M ←M – IMM. Branches relatively when M ≠ 0 (IMM = 0 to 31).	
	* DXBNE	X ←X – IMM. Branches relatively when X ≠ 0 (IMM = 0 to 31).	
	* DYBNE	Y ←Y – IMM. Branches relatively when Y ≠ 0 (IMM = 0 to 31).	
Return	RTI	PG, PC, PS ←Stack	
	RTL	PG, PC ←Stack	
	RTS	PC ←Stack	
Load & Return	* RTLD n	DPRn ←Stack, PG, PC ←Stack (n = 0 to 3. Multiple operations can be specified.)	
	* RTSD n	DPRn ←Stack, PC ←Stack (n = 0 to 3. Multiple operations can be specified.)	
Software interrupt	BRK	Generates a BRK interrupt.	
Special	STP	Stops oscillation.	
	WIT	Stops the CPU clock.	
No operation	NOP	PC ←PC + 1	

4.2 Description of each instruction

This section describes each instruction. Each instruction is described using one page per one instruction as a general rule. The description page is headed by the instruction mnemonic, and the pages are arranged in alphabetical order of the mnemonics. For each instruction, its operation and description **(Notes 1, 2)**, status flags' change, and a list sorted by addressing modes of the assembly language coding format **(Note 3)**, the machine code, the byte number and the minimum cycle number **(Note 4)** are described.

Notes 1: In the description of each instruction operation, the operation regarding PC (program counter) is described only for an instruction affecting the processing.
When an instruction is executed, its instruction bytes are added to the contents of PC and PC contains the address of the memory location of the instruction to be executed next. When a carry occurs at this addition, PG (program bank register) is incremented by 1.

2: [**Operation**] in the description of each instruction shows the contents of each register and memory after executing the instruction. The detailed operation sequence is omitted.

3: [**Description example**] in this manual is an example of assembly language description. Especially for addressing mode specification, various methods for mnemonic description in the 7900 Series are available, including the formats shown below. For more information, refer to the user's manual of the assembler to be used.

■ Methods for specifying addressing modes in Mitsubishi assembler

Addressing mode	Specification	Instruction coding example
Direct	DP0+:Offset6/8	ADD A,DP0+:04H
	DP0:label	ADD A,DP0:WORK
Direct indirect	(DP0+:Offset6/8)	ADD A,(DP0+:04H)
	(DP0:label)	ADD A,(DP0:WORK)
Direct indirect long	L(DP0+:Offset6/8)	ADD A,L(DP0+:04H)
	L(DP0:label)	ADD A,L(DP0:WORK)
Stack pointer relative	Offset,S	ADD A,05H,S
Stack pointer relative indirect indexed Y	(Offset,S),Y	ADD A,(05H,S),Y
Absolute	DT+:Offset16	ADD A,DT +:1000H
	DT:label	ADD A,DT:WORK
Absolute indirect	(Address)	JMP (1000H)
	(label)	JMP (TABLE)
Absolute long	LG:label	ADD A,LG:WORK
Absolute indirect long	L(DT+:Offset16)	ADD A,L(DT +:1000H)
	L(DT:label)	ADD A,L(DT:WORK)

- Offset6/8 : 6-bit offset value (when using DPR0 through DPR3) or 8-bit offset value (when using DPR0).
- Offset : 8-bit offset value.
- Offset16 : 16-bit offset value.
- Address : Memory address to be referenced.
- label : Label indicating the memory address to be referenced.

4.2 Description of each instruction

Notes 4: The cycle number shown is the minimum possible number, and this number depends on the following conditions:

- Value of direct page register's low-order byte

  The cycle number shown is a number when the direct page register's low-order byte (DPR_{nL}) is "00_{16}." When using an addressing mode that uses the direct page register in the condition of $DPR_{nL} \neq$ "00_{16}," the number which is obtained by adding 1 to the shown number is an actual cycle number.
- Number of bytes that have been loaded in an instruction queue buffer
- Whether the address of the memory read/write is even or odd
- Accessing of an external memory area in the condition of BYTE = "H" (using 8-bit external bus)
- Bus cycle

4.2 Description of each instruction

The following table shows the symbols that are used in instructions' description and the lists of this section, and each instruction is described bellow.

Symbol	Description
C	Carry flag
Z	Zero flag
I	Interrupt disable flag
D	Decimal mode flag
x	Index register length flag
m	Data length flag
V	Overflow flag
N	Negative flag
IPL	Processor interrupt priority level
+	Addition
–	Subtraction
×	Multiplication
*	Multiplication
÷	Division
/	Division
∧	Logical AND
∨	Logical OR
∀	Logical exclusive OR
| |	Absolute value
—	Negation
←	Movement toward the arrow direction
→	Movement toward the arrow direction
← →	Exchange
Acc	Accumulator
AccH	Accumulator's high-order 8 bits
AccL	Accumulator's low-order 8 bits
A	Accumulator A
AH	Accumulator A's high-order 8 bits
AL	Accumulator A's low-order 8 bits
B	Accumulator B
BH	Accumulator B's high-order 8 bits
BL	Accumulator B's low-order 8 bits
E	Accumulator E
EH	Accumulator E's high-order 16 bits
EL	Accumulator E's low-order 16 bits
X	Index register X
XH	Index register X's high-order 8 bits
XL	Index register X's low-order 8 bits
Y	Index register Y
YH	Index register Y's high-order 8 bits
YL	Index register Y's low-order 8 bits
S	Stack pointer

4.2 Description of each instruction

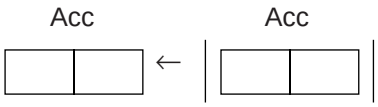
Symbol	Description
PC	Program counter
PC_H	Program counter's high-order 8 bits
PC_L	Program counter's low-order 8 bits
REL	Relative address
PG	Program bank register
DT	Data bank register
DPR0	Direct page register 0
$DPR0_H$	Direct page register 0's high-order 8 bits
$DPR0_L$	Direct page register 0's low-order 8 bits
DPRn	Direct page register n
$DPRn_H$	Direct page register n's high-order 8 bits
$DPRn_L$	Direct page register n's low-order 8 bits
PS	Processor status register
PS_H	Processor status register's high-order 8 bits
PS_L	Processor status register's low-order 8 bits
PS(bit n)	The n-th bit of processor status register
M	Memory contents
Mn, MEMn	n-bit address or contents of memory
M(oprd)	Contents of memory location specified by operand
M(bit n)	The n-th bit of the contents of memory
IMM	Immediate value (8 bits or 16 bits)
IMMn	n-bit immediate data
$IMMn_H$	High-order data of n-bit immediate data
$IMMn_L$	Low-order data of n-bit immediate data
EAR	Effective address (16 bits)
EAR_H	High-order 8 bits of effective address
EAR_L	Low-order 8 bits of effective address
MSB	Most significant bit
LSB	Least significant bit
dd	Displacement for DPR (8 bits or 6 bits)
$imm_{HH}imm_{HL}imm_{LH}imm_{LL}$	32-bit immediate value (bytes imm_{HH}, imm_{HL}, imm_{LH}, and imm_{LL} are shown from the highest one.)
imm_Himm_L	16-bit immediate value (imm_H represents the high-order 8 bits, and imm_L represents the low-order 8 bits.)
imm	8-bit immediate value
imm_n	n-bit immediate value
hhmmll	24-bit address value (hh represents the high-order 8 bits, mm represents the middle-order 8 bit, and ll represents the low-order 8 bits.)
mmll	16-bit address value (mm represents the high-order 8 bits, and ll represents the low-order 8 bits.)
nn	Displacement for S (8 bits)
n_1, n_2	8-bit data (2 types of 8-bit data)
rr	Displacement for PC (signed 8 bits)
rr_Hrr_L	Displacement for PC (signed 16 bits) (rr_H represents the high-order 8 bits, and rr_L represents the low-order 8 bits.)
hh_1, hh_2	Bank specification (2 types of 8-bit data)
source	Operand specified as transfer source
dest	Operand specified as transfer destination

Function : Absolute value

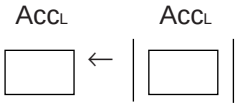
Operation data length: 16 bits or 8 bits

Operation : Acc ← |Acc|

When m = "0"



When m = "1"



❋ In this case, the contents of Acc_H do not change.

Description : Obtains the absolute value of Acc contents and stores the result in Acc.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	V	—	—	—	—	Z	0

N : Always "0" because MSB of the operation result is "0."

V : Set to "1" if the operation result exceeds +32767 (or +127 when m = "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Always "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ABS A	$E1_{16}$	1	3
A	ABS B	81_{16}, $E1_{16}$	2	4

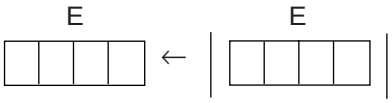
Description example:

```
CLM
ABS     A               ; A ← |A|
SEM
ABS     B               ; BL ← |BL|
```

Function : Absolute value

Operation data length: 32 bits

Operation : E ← |E|



Description : Obtains the absolute value of the E contents and stores the result in E.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	V	—	—	—	—	Z	0

N : Always “0” because MSB of the operation result is “0.”

V : Set to “1” if the operation result exceeds +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Always “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ABSD E	31_{16}, 90_{16}	2	5

Description example:

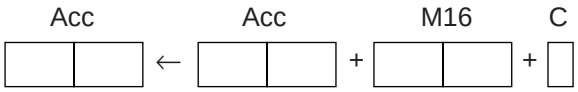
```
ABSD    E                ; E ← |E|
```

Function : Addition with carry

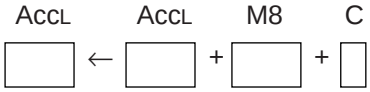
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc + M + C

When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Adds the contents of Acc, memory, and flag C, and stores the result in Acc.

● This instruction operates in decimal when flag D = “1.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.” Meaningless when flag D = “1.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m = “1”). Otherwise, cleared to “0.” Meaningless when flag D = “1.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.” Meaningless when flag D = “1.”

C : Set to “1” when flag D = “0” and the result of the operation (regarded as an unsigned operation) exceeds +65535 (+255 when flag m = “1”). Otherwise, cleared to “0.”

Set to “1” when flag D = “1” and the result of the operation (regarded as an unsigned operation) exceeds +9999 (+99 when flag m = “1”). Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADC A, #imm	31_{16}, 87_{16}, imm ($B1_{16}$, 87_{16}, imm)	3	3 (3)
DIR	ADC A, dd	21_{16}, $8A_{16}$, dd ($A1_{16}$, $8A_{16}$, dd)	3	5 (7)
DIR, X	ADC A, dd, X	21_{16}, $8B_{16}$, dd ($A1_{16}$, $8B_{16}$, dd)	3	6 (8)
(DIR)	ADC A, (dd)	21_{16}, 80_{16}, dd ($A1_{16}$, 80_{16}, dd)	3	7 (9)
(DIR, X)	ADC A, (dd, X)	21_{16}, 81_{16}, dd ($A1_{16}$, 81_{16}, dd)	3	8 (10)
(DIR), Y	ADC A, (dd), Y	21_{16}, 88_{16}, dd ($A1_{16}$, 88_{16}, dd)	3	8 (10)
L(DIR)	ADC A, L(dd)	21_{16}, 82_{16}, dd ($A1_{16}$, 82_{16}, dd)	3	9 (11)
L(DIR), Y	ADC A, L(dd), Y	21_{16}, 89_{16}, dd ($A1_{16}$, 89_{16}, dd)	3	10(12)
SR	ADC A, nn, S	21_{16}, 83_{16}, nn ($A1_{16}$, 83_{16}, nn)	3	6 (8)
(SR), Y	ADC A, (nn, S), Y	21_{16}, 84_{16}, nn ($A1_{16}$, 84_{16}, nn)	3	9 (11)
ABS	ADC A, mmll	21_{16}, $8E_{16}$, ll, mm ($A1_{16}$, $8E_{16}$, ll, mm)	4	5 (7)
ABS, X	ADC A, mmll, X	21_{16}, $8F_{16}$, ll, mm ($A1_{16}$, $8F_{16}$, ll, mm)	4	6 (8)
ABS, Y	ADC A, mmll, Y	21_{16}, 86_{16}, ll, mm ($A1_{16}$, 86_{16}, ll, mm)	4	6 (8)
ABL	ADC A, hhmmll	21_{16}, $8C_{16}$, ll, mm, hh ($A1_{16}$, $8C_{16}$, ll, mm, hh)	5	6 (8)
ABL, X	ADC A, hhmmll, X	21_{16}, $8D_{16}$, ll, mm, hh ($A1_{16}$, $8D_{16}$, ll, mm, hh)	5	7 (9)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace “A” with “B” in the syntax. In this case, the machine code and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = “0.”

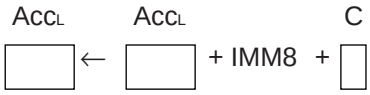
Description example:

```
CLM
ADC.W   A, #IMM16       ; A ← A + IMM16 + C
ADC     B, MEM16        ; B ← B + MEM16 + C
SEM
ADC.B   A, #IMM8        ; AL ← AL + IMM8 + C
ADC     B, MEM8         ; BL ← BL + MEM8 + C
```

Function : Addition with carry

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L + IMM8 + C$



Description : Adds the contents of Acc_L, the immediate value, and flag C in 8-bit length, and stores the result in Acc_L.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.
- This instruction operates in decimal when flag D = "1."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0." Meaningless when flag D = "1."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to "0." Meaningless when flag D = "1."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0." Meaningless when flag D = "1."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +255 (+99 when flag D = "1"). Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADCB A, #imm	31_{16}, $1A_{16}$, imm	3	3
IMM	ADCB B, #imm	$B1_{16}$, $1A_{16}$, imm	3	3

Description example:

```
ADCB    A, #IMM8        ; AL ← AL + IMM8 + C
ADCB    B, #IMM8        ; BL ← BL + IMM8 + C
```

Function : Addition with carry

Operation data length: 32 bits

Operation : E ← E + M32 + C

Description : Adds contents of E, memory, and flag C in 32-bit length, and stores the result in E. CPU operates as binary addition in spite of the contents of decimal mode flag.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +4294967295. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADCD E, #imm	31_{16}, $1C_{16}$, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	6	4
DIR	ADCD E, dd	21_{16}, $9A_{16}$, dd	3	7
DIR, X	ADCD E, dd, X	21_{16}, $9B_{16}$, dd	3	8
(DIR)	ADCD E, (dd)	21_{16}, 90_{16}, dd	3	9
(DIR, X)	ADCD E, (dd, X)	21_{16}, 91_{16}, dd	3	10
(DIR), Y	ADCD E, (dd), Y	21_{16}, 98_{16}, dd	3	10
L(DIR)	ADCD E, L(dd)	21_{16}, 92_{16}, dd	3	11
L(DIR), Y	ADCD E, L(dd), Y	21_{16}, 99_{16}, dd	3	12
SR	ADCD E, nn, S	21_{16}, 93_{16}, nn	3	8
(SR), Y	ADCD E, (nn, S), Y	21_{16}, 94_{16}, nn	3	11
ABS	ADCD E, mmll	21_{16}, $9E_{16}$, ll, mm	4	7
ABS, X	ADCD E, mmll, X	21_{16}, $9F_{16}$, ll, mm	4	8
ABS, Y	ADCD E, mmll, Y	21_{16}, 96_{16}, ll, mm	4	8
ABL	ADCD E, hhmmll	21_{16}, $9C_{16}$, ll, mm, hh	5	8
ABL, X	ADCD E, hhmmll, X	21_{16}, $9D_{16}$, ll, mm, hh	5	9

Description example:

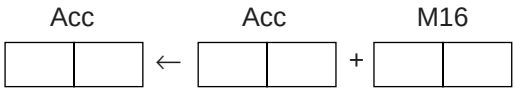
```
ADCD    E, #IMM32       ; E ← E + IMM32 + C
                        ; (B, A ← B, A + IMM32 + C)
ADCD    E, MEM32        ; E ← E + MEM32 + C
                        ; (B, A ← B, A + MEM32 + C)
```

Function : Addition

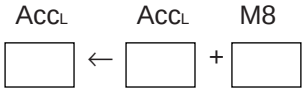
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc + M

When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Adds the contents of Acc and memory, and stores the result in Acc.

● This instruction cannot operate in decimal. Clear flag D to “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m = “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when the result of the operation (regarded as an unsigned operation) exceeds +65535 (+255 when flag m = “1”). Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADD A, #imm	26_{16}, imm (81_{16}, 26_{16}, imm)	2 (3)	1 (2)
DIR	ADD A, dd	$2A_{16}$, dd (81_{16}, $2A_{16}$, dd)	2 (3)	3 (4)
DIR, X	ADD A, dd, X	$2B_{16}$, dd (81_{16}, $2B_{16}$, dd)	2 (3)	4 (5)
(DIR)	ADD A, (dd)	11_{16}, 20_{16}, dd (91_{16}, 20_{16}, dd)	3 (3)	6 (6)
(DIR, X)	ADD A, (dd, X)	11_{16}, 21_{16}, dd (91_{16}, 21_{16}, dd)	3 (3)	7 (7)
(DIR), Y	ADD A, (dd), Y	11_{16}, 28_{16}, dd (91_{16}, 28_{16}, dd)	3 (3)	7 (7)
L(DIR)	ADD A, L(dd)	11_{16}, 22_{16}, dd (91_{16}, 22_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	ADD A, L(dd), Y	11_{16}, 29_{16}, dd (91_{16}, 29_{16}, dd)	3 (3)	9 (9)
SR	ADD A, nn, S	11_{16}, 23_{16}, nn (91_{16}, 23_{16}, nn)	3 (3)	5 (5)
(SR), Y	ADD A, (nn, S), Y	11_{16}, 24_{16}, nn (91_{16}, 24_{16}, nn)	3 (3)	8 (8)
ABS	ADD A, mmll	$2E_{16}$, ll, mm (81_{16}, $2E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	ADD A, mmll, X	$2F_{16}$, ll, mm (81_{16}, $2F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	ADD A, mmll, Y	11_{16}, 26_{16}, ll, mm (91_{16}, 26_{16}, ll, mm)	4 (4)	5 (5)
ABL	ADD A, hhmmll	11_{16}, $2C_{16}$, ll, mm, hh (91_{16}, $2C_{16}$, ll, mm, hh)	5 (5)	5 (5)
ABL, X	ADD A, hhmmll, X	11_{16}, $2D_{16}$, ll, mm, hh (91_{16}, $2D_{16}$, ll, mm, hh)	5 (5)	6 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace “A” with “B” in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = “0.”

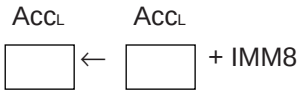
Description example:

```
CLM
ADD.W    A, #IMM16          ; A ← A + IMM16
ADD      B, MEM16           ; B ← B + MEM16
SEM
ADD.B    A, #IMM8           ; AL ← AL + IMM8
ADD      B, MEM8            ; BL ← BL + MEM8
```

Function : Addition

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L + IMM8$



Description : Adds the contents of Acc_L and immediate value in 8-bit length, and stores the result in Acc_L.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +255. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADDB A, #imm	29_{16}, imm	2	1
IMM	ADDB B, #imm	81_{16}, 29_{16}, imm	3	2

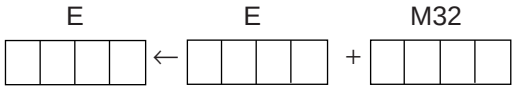
Description example:

```
ADDB    A, #IMM8        ; AL ← AL + IMM8
ADDB    B, #IMM8        ; BL ← BL + IMM8
```

Function : Addition

Operation data length: 32 bits

Operation : E ← E + M32



Description : Adds the contents of E and memory in 32-bit length, and stores the result in the E.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +4294967295. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADDD E, #imm	$2D_{16}$, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	5	3
DIR	ADDD E, dd	$9A_{16}$, dd	2	6
DIR, X	ADDD E, dd, X	$9B_{16}$, dd	2	7
(DIR)	ADDD E, (dd)	11_{16}, 90_{16}, dd	3	9
(DIR, X)	ADDD E, (dd, X)	11_{16}, 91_{16}, dd	3	10
(DIR), Y	ADDD E, (dd), Y	11_{16}, 98_{16}, dd	3	10
L(DIR)	ADDD E, L(dd)	11_{16}, 92_{16}, dd	3	11
L(DIR), Y	ADDD E, L(dd), Y	11_{16}, 99_{16}, dd	3	12
SR	ADDD E, nn, S	11_{16}, 93_{16}, nn	3	8
(SR), Y	ADDD E, (nn, S), Y	11_{16}, 94_{16}, nn	3	11
ABS	ADDD E, mmll	$9E_{16}$, ll, mm	3	6
ABS, X	ADDD E, mmll, X	$9F_{16}$, ll, mm	3	7
ABS, Y	ADDD E, mmll, Y	11_{16}, 96_{16}, ll, mm	4	8
ABL	ADDD E, hhmmll	11_{16}, $9C_{16}$, ll, mm, hh	5	8
ABL, X	ADDD E, hhmmll, X	11_{16}, $9D_{16}$, ll, mm, hh	5	9

Description example:

```
ADDD    E, #IMM32      ; E ← E + IMM32 (B, A ← B, A + IMM32)
ADDD    E, MEM32       ; E ← E + MEM32 (B, A ← B, A + MEM32)
```

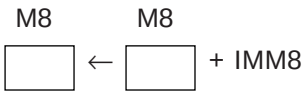
Function : Addition

Operation data length: 16 bits or 8 bits

Operation : M← M + IMM

When m = “0”

When m = “1”



Description : Adds the contents of memory and immediate value, and stores the result in memory.

- This instruction cannot operate in decimal. Clear flag D to “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m = “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the result of the operation is “0.” Otherwise, cleared to “0.”

C : Set to “1” when the result of the operation (regarded as an unsigned operation) exceeds +65535 (+255 when flag m = “1”). Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ADDM dd, #imm	51_{16}, 03_{16}, dd, imm	4	7
ABS	ADDM mmll, #imm	51_{16}, 07_{16}, ll, mm, imm	5	7

Note : When flag m = “0,” the byte number increases by 1.

Description example:

```
CLM
ADDM.W    MEM16, #IMM16        ; MEM16 ← MEM16 + IMM16
SEM
ADDM.B    MEM8, #IMM8          ; MEM8 ← MEM8 + IMM8
```

Function : Addition

Operation data length: 8 bits

Operation : M8 ← M8 + IMM8

M8 M8

□ ← □ + IMM8

Description : Adds the contents of memory and immediate value in 8-bit length, and stored the result in memory.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +255. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ADDMB dd, #imm	51_{16}, 02_{16}, dd, imm	4	7
ABS	ADDMB mmll, #imm	51_{16}, 06_{16}, ll, mm, imm	5	7

Description example:

```
ADDMB    MEM8, #IMM8          ; MEM8 ← MEM8 + IMM8
```

Function : Addition

Operation data length: 32 bits

Operation : M32 ← M32 + IMM32

Description : Adds the contents of memory and immediate value in 32-bit length, and stores the result in memory.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when the result of the operation (regarded as an unsigned operation) exceeds +4294967295. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ADDMD dd, #imm	51_{16}, 83_{16}, dd, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	7	10
ABS	ADDMD mmll, #imm	51_{16}, 87_{16}, ll, mm, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	8	10

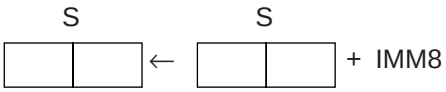
Description example:

```
ADDMD   MEM32, #IMM32          ; MEM32 ← MEM32 + IMM32
```

Function : Addition

Operation data length: 16 bits

Operation : S ← S + IMM8



Description : Adds the contents of S and 8-bit immediate value in 16-bit length, and stores the result in S. Extend zero of the immediate value to the 16-bit immediate value, at the operation.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +65535. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADDS #imm	31_{16}, $0A_{16}$, imm	3	2

Description example:

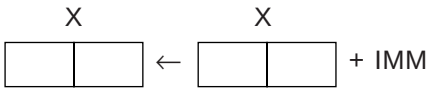
```
ADDS    #IMM8                   ; S ← S + IMM8
```

Function : Addition

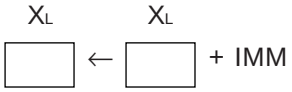
Operation data length: 16 bits or 8 bits

Operation : X← X + IMM (IMM = 0 to 31)

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Adds the contents of X and immediate value (0 to 31), and stores the result in X.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag x is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +65535 (+255 when flag x = "1"). Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADDX #imm	01_{16}, imm	2	2

Note : Any value from 0 to 31 can be set to imm.

Description example:

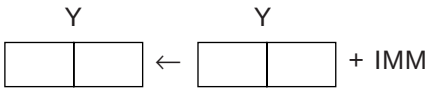
```
CLP     x
ADDX    #IMM            ; X ← X + IMM
SEP     x
ADDX    #IMM            ; XL ← XL + IMM
```

Function : Addition

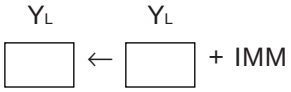
Operation data length: 16 bits or 8 bits

Operation : Y← Y + IMM (IMM = 0 to 31)

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Adds the contents of Y and immediate value (0 to 31), and stores the result in Y.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Clear flag D to "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag x is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" when the result of the operation (regarded as an unsigned operation) exceeds +65535 (+255 when flag x = "1"). Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ADDY #imm	01_{16}, imm+20_{16}	2	2

Note : Any value from 0 to 31 can be set to imm.

Description example:

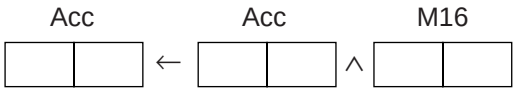
```
CLP     x
ADDX    #IMM            ; Y ← Y + IMM
SEP     x
ADDX    #IMM            ; YL ← YL + IMM
```

Function : Logical AND

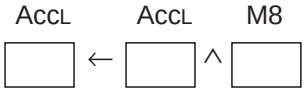
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc ∧ M

When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Performs logical AND between the contents of Acc and the contents of a memory, and stores the result in Acc.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	AND A, #imm	66_{16}, imm (81_{16}, 66_{16}, imm)	2 (3)	1 (2)
DIR	AND A, dd	$6A_{16}$, dd (81_{16}, $6A_{16}$, dd)	2 (3)	3 (4)
DIR, X	AND A, dd, X	$6B_{16}$, dd (81_{16}, $6B_{16}$, dd)	2 (3)	4 (5)
(DIR)	AND A, (dd)	11_{16}, 60_{16}, dd (91_{16}, 60_{16}, dd)	3 (3)	6 (6)
(DIR, X)	AND A, (dd, X)	11_{16}, 61_{16}, dd (91_{16}, 61_{16}, dd)	3 (3)	7 (7)
(DIR), Y	AND A, (dd), Y	11_{16}, 68_{16}, dd (91_{16}, 68_{16}, dd)	3 (3)	7 (7)
L(DIR)	AND A, L(dd)	11_{16}, 62_{16}, dd (91_{16}, 62_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	AND A, L(dd), Y	11_{16}, 69_{16}, dd (91_{16}, 69_{16}, dd)	3 (3)	9 (9)
SR	AND A, nn, S	11_{16}, 63_{16}, nn (91_{16}, 63_{16}, nn)	3 (3)	5 (5)
(SR), Y	AND A, (nn, S), Y	11_{16}, 64_{16}, nn (91_{16}, 64_{16}, nn)	3 (3)	8 (8)
ABS	AND A, mmll	$6E_{16}$, ll, mm (81_{16}, $6E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	AND A, mmll, X	$6F_{16}$, ll, mm (81_{16}, $6F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	AND A, mmll, Y	11_{16}, 66_{16}, ll, mm (91_{16}, 66_{16}, ll, mm)	4 (4)	5 (5)
ABL	AND A, hhmmll	11_{16}, $6C_{16}$, ll, mm, hh (91_{16}, $6C_{16}$, ll, mm, hh)	5 (5)	5 (5)
ABL, X	AND A, hhmmll, X	11_{16}, $6D_{16}$, ll, mm, hh (91_{16}, $6D_{16}$, ll, mm, hh)	5 (5)	6 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace "A" with "B" in the syntax. In this case, the machine code,the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

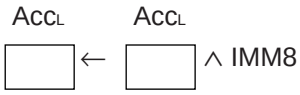
Description example:

```
CLM
AND.W    A, #IMM16      ; A ← A ∧ IMM16
AND      B, MEM16       ; B ← B ∧ MEM16
SEM
AND.B    A, #IMM8       ; AL ← AL ∧ IMM8
AND      B, MEM8        ; BL ← BL ∧ MEM8
```

Function : Logical AND

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L \wedge IMM8$



Description : Performs logical AND between the contents of Acc_L and the immediate value in 8-bit length, and stores the result in Acc_L.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ANDB A, #imm	23_{16}, imm	2	1
IMM	ANDB B, #imm	81_{16}, 23_{16}, imm	3	2

Description example:

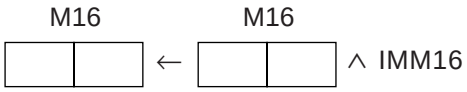
```
ANDB    A, #IMM8        ; AL ← AL ∧ IMM8
ANDB    B, #IMM8        ; BL ← BL ∧ IMM8
```

Function : Logical AND

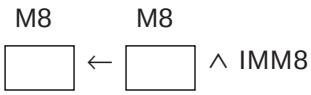
Operation data length: 16 bits or 8 bits

Operation : M← M ∧ IMM

When m = “0”



When m = “1”



Description : Performs logical AND between the contents of memory and immediate value, and stores the result in the memory.

● This instruction includes the function of the CLB instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ANDM dd, #imm	51_{16}, 63_{16}, dd, imm	4	7
ABS	ANDM mmll, #imm	51_{16}, 67_{16}, ll, mm, imm	5	7

Note : When flag m = “0,” the byte number increases by 1.

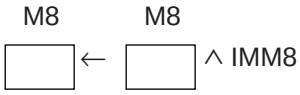
Description example:

```
CLM
ANDM.W    MEM16, #IMM16        ; MEM16 ← MEM16 ∧ IMM16
SEM
ANDM.B    MEM8, #IMM8          ; MEM8 ← MEM8 ∧ IMM8
```

Function : Logical AND

Operation data length: 8 bits

Operation : M8 ← M8 ∧ IMM8



Description : Performs logical AND between the contents of memory and immediate value in 8-bit length, and stores the result in the memory.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ANDMB dd, #imm	51_{16}, 62_{16}, dd, imm	4	7
ABS	ANDMB mmll, #imm	51_{16}, 66_{16}, ll, mm, imm	5	7

Description example:

```
ANDMB    MEM8, #IMM8        ; MEM8 ← MEM8 ∧IMM8
```

ANDMD logical AND between immediate value and Memory (Double word) ANDMD

Function : Logical AND

Operation data length: 32 bits

Operation : M32 ← M32 ∧ IMM32

Description : Performs logical AND between the contents of memory and immediate value in 32-bit length, and stores the result in the memory.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ANDMD dd, #imm	51_{16}, $E3_{16}$, dd, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	7	10
ABS	ANDMD mmll, #imm	51_{16}, $E7_{16}$, ll, mm, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	8	10

Description example:

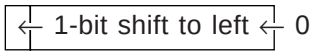
```
ANDMD    MEM32, #IMM32        ; MEM32 ← MEM32 ∧IMM32
```

Function : Arithmetic shift to the left

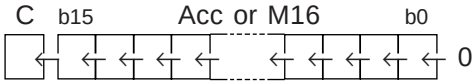
Operation data length: 16 bits or 8 bits

Operation :

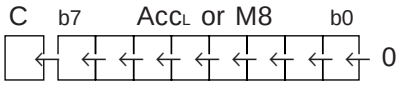
C Acc or M



When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Shifts all bits of Acc or a memory to left by 1 bit. In this time, a “0” is placed in LSB of Acc or the memory. MSB before the shift is placed in flag C.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when MSB of Acc or the memory before the operation is “1.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ASL A	03_{16}	1	1
A	ASL B	81_{16}, 03_{16}	2	2
DIR	ASL dd	21_{16}, $0A_{16}$, dd	3	7
DIR, X	ASL dd, X	21_{16}, $0B_{16}$, dd	3	8
ABS	ASL mmll	21_{16}, $0E_{16}$, ll, mm	4	7
ABS, X	ASL mmll, X	21_{16}, $0F_{16}$, ll, mm	4	8

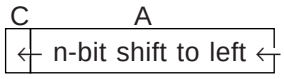
Description example:

```
CLM
ASL     A       ; A ← A is arithmetically shifted left by 1 bit.
ASL     MEM16   ; MEM16 ← MEM16 is arithmetically shifted left by 1 bit.
SEM
ASL     A       ; AL ← AL is arithmetically shifted left by 1 bit.
ASL     MEM8    ; MEM8 ← MEM8 is arithmetically shifted left by 1 bit.
```

Function : Arithmetic shift to the left

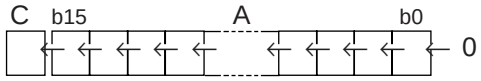
Operation data length: 16 bits or 8 bits

Operation :

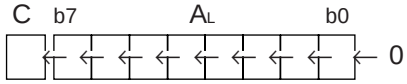


(n : Number of times shifted. n = 0 to 15)

When m = "0"



When m = "1"



❋ In this case, the contents of A_H do not change.

Description : Shifts all bits of A to the left by n bits. In this case, a "0" is placed in bit 0 of A each time its contents are shifted by 1 bit. MSB is placed in flag C each time its contents are shifted by 1 bit.

● B cannot be used in this instruction.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" if MSB = "1" when the contents are shifted by (n – 1) bits. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ASL A, #imm	$C1_{16}$, imm+40_{16}	2	imm+6

Note : Any value (number of times shifted) from 0 to 15 can be set to imm.

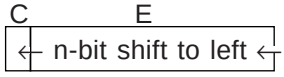
Description example:

```
CLM
ASL     A, #15      ; A ← A is arithmetically shifted to the left by 15 bits.
SEM
ASL     A, #7       ; AL ← AL is arithmetically shifted to the left by 7 bits.
```

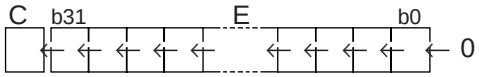
Function : Arithmetic shift to the left

Operation data length: 32 bits

Operation : C E



(n : Number of times shifted. n = 0 to 31)



Description : Shifts all bits of E in 32-bit length to the left by n bits. In this case, a "0" is placed in bit 0 of E each time its contents are shifted by 1 bit. MSB is placed in flag C each time its contents are shifted by 1 bit.

- This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" if MSB = "1" when the contents are shifted by (n – 1) bits. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ASLD E, #imm	$D1_{16}$, imm+40_{16}	2	imm+8

Note : Any value (number of times shifted) from 0 to 31 can be set to imm.

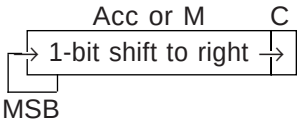
Description example:

```
ASLD    E, #16          ; E ← E is arithmetically shifted to the left by 16 bits.
```

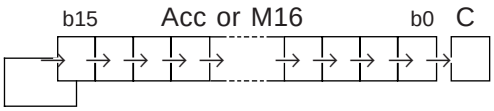
Function : Arithmetic shift to the right

Operation data length: 16 bits or 8 bits

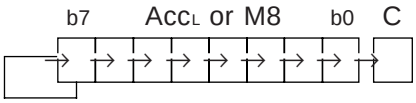
Operation :



When m = “0”



When m = “1”



❋ In this case, the contents of Acc$_H$ do not change.

Description : Shifts all bits of Acc or a memory to the left by 1 bit. In this time, MSB before the shift is placed in MSB of Acc or the memory. LSB before the shift is placed in LSB.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when LSB of Acc or the memory before the operation is “1.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ASR A	64_{16}	1	1
A	ASR B	81_{16}, 64_{16}	2	2
DIR	ASR dd	21_{16}, $4A_{16}$, dd	3	7
DIR, X	ASR dd, X	21_{16}, $4B_{16}$, dd	3	8
ABS	ASR mmll	21_{16}, $4E_{16}$, ll, mm	4	7
ABS, X	ASR mmll, X	21_{16}, $4F_{16}$, ll, mm	4	8

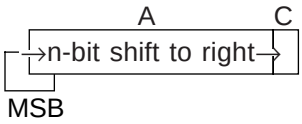
Description example:

```
CLM
ASR     A       ; A ← A is arithmetically shifted to the right by 1 bit.
ASR     MEM16   ; MEM16 ← MEM16 is arithmetically shifted to the right by 1 bit.
SEM
ASR     A       ; AL ← AL is arithmetically shifted to the right by 1 bit.
ASR     MEM8    ; MEM8 ← MEM8 is arithmetically shifted to the right by 1 bit.
```

Function : Arithmetic shift to the right

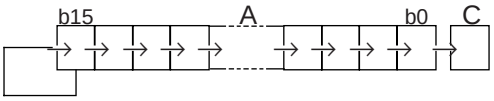
Operation data length: 16 bits or 8 bits

Operation :

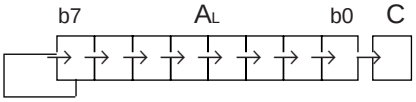


(n : Number of times shifted. n = 0 to 15)

When m = "0"



When m = "1"



❋ In this case, the contents of A_H do not change.

Description : Shifts all bits of A to the right by n bits. In this time, MSB before the shift is placed in MSB of A. LSB is placed in flag C each time its contents are shifted by 1 bit.

● B cannot be used in this instruction.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" if LSB = "1" when the contents are shifted by (n – 1) bits. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ASR A, #imm	$C1_{16}$, imm+80_{16}	2	imm+6

Note : Any value (number of times shifted) from 0 to 15 can be set to imm.

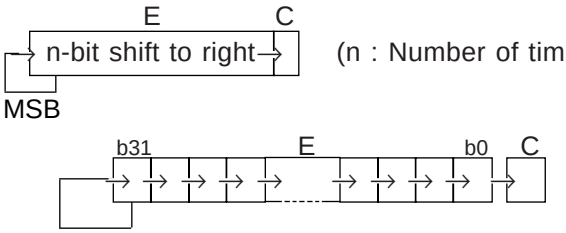
Description example:

```
CLM
ASR     A, #15      ; A ← A is arithmetically shifted to the right by 15 bits.
SEM
ASR     A, #7       ; AL ← AL is arithmetically shifted to the right by 7 bits.
```

Function : Arithmetic shift to the right

Operation data length: 32 bits

Operation :



(n : Number of times shifted. n = 0 to 31)

Description : Shifts all bits of E in 32-bit length to the right by n bits. In this time, MSB before the shift is placed in MSB of E. LSB is placed in flag C each time its contents are shifted by 1 bit.

● This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” if LSB = “1” when the contents are shifted by (n – 1) bits. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ASRD E, #imm	$D1_{16}$, imm+80_{16}	2	imm+8

Note : Any value (number of times shifted) from 0 to 31 can be set to imm.

Description example:

```
ASRD    E, #16      ; E ← E is arithmetically shifted to the right by 16 bits.
```

Function : Conditional branch

Operation data length: 16 bits or 8 bits

Operation : Relative branch to the specified address when M (bit n) = “0” (n specifies a bit position; multiple bits can be specified.)

Description : Branches to the specified address if the contents of all specified bits in memory (multiple bits can be specified) are “0”s. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address. The bit positions to be tested are indicated by a bit pattern of the immediate value, in which the bits set to “1” are the subject bits to be tested.

- When m=“0” : This instruction operates in 16-bit length.
  When m=“1” : This instruction operates in 8-bit length.
- Branches when no bit is specified that need to be tested.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR, b, R	BBC #imm, dd, rr	41_{16}, $5A_{16}$, dd, imm, rr	5	9
ABS, b, R	BBC #imm, mmll, rr	41_{16}, $5E_{16}$, ll, mm, imm, rr	6	9

Note : When flag m = “0,” the byte number increases by 1.

Description example:

```
CLM
BBC.W   #IMM16, MEM16, LABEL1   ; Branches to LABEL1 if all specified bits in MEM16 are “0”s.
SEM
BBC.B   #IMM8, MEM8, LABEL2     ; Branches to LABEL2 if all specified bits in MEM8 are “0”s.
```

Function : Conditional branch

Operation data length: 8 bits

Operation : Relative branch to the specified address when M8 (bit n) = “0” (n specifies a bit position; multiple bits can be specified.)

Description : Branches to the specified address if the contents of all specified bits in memory (multiple bits can be specified) are “0”s. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address. The bit positions to be tested are indicated by a bit pattern of the 8-bit immediate value, in which the bits set to “1” are the subject bits to be tested.

- Branches if no bit is specified that need to be tested.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR, b, R	BBCB #imm, dd, rr	52_{16}, dd, imm, rr	4	8
ABS, b, R	BBCB #imm, mmll, rr	57_{16}, ll, mm, imm, rr	5	8

Description example:

```
BBCB    #IMM8, MEM8, LABEL    ; Branches to LABEL if all specified bits in MEM8 are 0s.
```

Function : Conditional branch

Operation data length: 16 bits or 8 bits

Operation : Relative branch to the specified address when M (bit n) = “1” (n specifies a bit position; multiple bits can be specified.)

Description : Branches to the specified address if the contents of all specified bits in memory (multiple bits can be specified) are “1”s. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address. The bit positions to be tested are indicated by a bit pattern of the immediate value, in which the bits set to “1” are the subject bits to be tested.

- When m=“0” : This instruction operates in 16-bit length.
  When m=“1” : This instruction operates in 8-bit length.
- Branches if no bit is specified that need to be tested.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR, b, R	BBS #imm, dd, rr	41_{16}, $4A_{16}$, dd, imm, rr	5	9
ABS, b, R	BBS #imm, mmll, rr	41_{16}, $4E_{16}$, ll, mm, imm, rr	6	9

Note : When flag m = “0,” the byte number increases by 1.

Description example:

```
CLM
BBS.W   #IMM16, MEM16, LABEL1   ; Branches to LABEL1 if all specified bits in MEM16 are “1”s.
SEM
BBS.B   #IMM8, MEM8, LABEL2     ; Branches to LABEL2 if all specified bits in MEM8 are “1”s.
```

Function : Conditional branch

Operation data length: 8 bits

Operation : Relative branch to the specified address when M8 (bit n) = “1” (n specifies a bit position; multiple bits can be specified.)

Description : Branches to the specified address if the contents of all specified bits in memory (multiple bits can be specified) are “1”s. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address. The bit positions to be tested are indicated by a bit pattern of the 8-bit immediate value, in which the bits set to “1” are the subject bits to be tested.

- Branches if no bit is specified that need to be tested.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR, b, R	BBSB #imm, dd, rr	42_{16}, dd, imm, rr	4	8
ABS, b, R	BBSB #imm, mmll, rr	47_{16}, ll, mm, imm, rr	5	8

Description example:

```
BBSB    #IMM8, MEM8, LABEL    ; Branches to LABEL if all specified bits in MEM8 are “1”s.
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when C = “0.”

Description : Branches to the specified address if flag C is “0.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BCC rr	90_{16}, rr	2	6

Description example:

```
BCC          LABEL          ; Branches to LABEL if C = “0.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when C = “1.”

Description : Branches to the specified address if flag C is “1.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BCS rr	$B0_{16}$, rr	2	6

Description example:

```
BCS     LABEL     ; Branches to LABEL if C = “1.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when Z = “1.”

Description : Branches to the specified address if flag Z is “1.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BEQ rr	$F0_{16}$, rr	2	6

Description example:

```
BEQ     LABEL     ; Branches to LABEL if Z = “1.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when N∀V = "0."

Description : Branches to the specified address if the contents of flags N and V are the same. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

- Branches when the result of the compare instruction or the subtract instruction satisfies "Greater or Equal ≥" condition.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BGE rr	$C0_{16}$, rr	2	6

Description example:

```
BGE     LABEL     ; Branches to LABEL if N∀V = "0."
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when Z = “0” and N∀V = “0.”

Description : Branches to the specified address if flag Z is “0” and the contents of flags N and V are the same. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

- Branches when the result of the compare instruction or the subtract instruction satisfies signed “Greater than >” condition.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BGT rr	80_{16}, rr	2	6

Description example:

```
BGT     LABEL          ; Branches to LABEL if Z = “0” and N∀V = “0.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to specified address if C = “1” and flag Z = “0.”

Description : Branches to the specified address if flag C is “1” and flag Z is “0.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

- Branches when the result of the compare instruction or the subtract instruction satisfies unsigned “Greater than >” condition.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BGTU rr	40_{16}, rr	2	6

Description example:

```
BGTU    LABEL    ; Branches to LABEL if C = “1” and Z = “0.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to specified address when Z = “1” or N∀V = “1.”

Description : Branches to the specified address if flag Z is “1” or the contents of flags N and V are different. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

- Branches when the result of the compare instruction or the subtract instruction satisfies signed “Less or Equal ≤” condition.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BLE rr	$A0_{16}$, rr	2	6

Description example:

```
BLE     LABEL     ; Branches to LABEL if Z= “1” and N∀V = “1.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address if C = “0” or Z = “1.”

Description : Branches to the specified address if flag C is “0” or flag Z is “1.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

- Branches when the result of the compare instruction or the subtract instruction satisfies unsigned “Less or Equal ≤” condition.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BLEU rr	60_{16}, rr	2	6

Description example:

```
BLEU        LABEL          ; Branches to LABEL if C = “0” or Z = “1.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to specified address when N∀V = “1.”

Description : Branches to the specified address if the contents of flags N and V are different. Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

- Branches when the result of the compare instruction or the subtract instruction satisfies “Less than <” condition.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BLT rr	$E0_{16}$, rr	2	6

Description example:

```
BLT     LABEL     ; Branches to LABEL if N∀V = “1.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to specified address if N = “1.”

Description : Branches to the specified address if flag N is “1.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BMI rr	30_{16}, rr	2	6

Description example:

```
BMI     LABEL     ; Branches to LABEL if N = “1.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address if Z = “0.”

Description : Branches to the specified address if flag Z is “0.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BNE rr	$D0_{16}$, rr	2	6

Description example:

```
BNE     LABEL     ; Branches to LABEL if Z = “0.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when N = “0.”

Description : Branches to the specified address if flag N is “0.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BPL rr	10_{16}, rr	2	6

Description example:

```
BPL     LABEL     ; Branches to LABEL if N = “0.”
```

Function : Unconditional branch

Operation data length: –

Operation : PC ← PC + cnt + REL (cnt : byte number of the BRA/BRAL instruction)

Description : Branches always to the specified address. Use an 8-bit value relative to PC (BRA : –128 to +127) or a 16-bit value relative to PC (BRAL : –32768 to +32767) after the branch instruction execution to specify the branch destination address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BRA rr	20_{16}, rr	2	5
	BRAL $rr_H rr_L$	$A7_{16}$, rr_L, rr_H	3	5

Description example:

```
BRA     REL8     ; Branches to address (PC + 2 + REL8)
BRAL    REL16    ; Branches to address (PC + 3 + REL16)
```

Function : Software interrupt

Operation data length: –

Operation : Generate a BRK interrupt

Description : Saves the address where the instruction next to the BRK instruction is stored and the PS contents in order of PG, PC, and PS to the stack. Then, branches to the address whose low-order address is the contents of address $FFFA_{16}$ and high-order address is the contents of address $FFFB_{16}$.

- This instruction is reserved for use in debug tools and cannot be used when using an emulator.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	1	—	—

I : Set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	BRK	00_{16}, 74_{16}	2	15

Description example:

```
BRK                    ;
```

Function : Conditional branch

Operation data length: 16 bits or 8 bits

Operation : Relative branch to the specified address when A (bit n) = 0 or M (bit n) = 0 (n = 0 to 15. Only 1 bit can be specified).

Description : Branches to the specified address if the contents of the specified bit of A or a memory is "0." Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- When m = "0" : Any 1 bit between b0 to b15 can be specified.
  When m = "1" : Any 1 bit between b0 to b7 can be specified.
- B cannot be used in this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	BSC n, A, rr	01_{16}, n+$A0_{16}$, rr	3	7
DIR	BSC n, dd, rr	71_{16}, n+$A0_{16}$, dd, rr	4	11
ABS	BSC n, mmll, rr	71_{16}, n+$E0_{16}$, ll, mm, rr	5	10

Note : Any value from 0 to 15 can be set to n.

Description example:

```
CLM
BSC     8, A, LABEL1        ; Branches to LABEL1 if b8 of A is "0."
BSC     15, MEM16, LABEL2   ; Branches to LABEL2 if b15 of MEM16 is "0."
SEM
BSC     7, A, LABEL3        ; Branches to LABEL3 if b7 of A is "0."
BSC     7, MEM8, LABEL4     ; Branches to LABEL4 if b7 of MEM8 is "0."
```

Function : Subroutine call

Operation data length: –

Operation : Stack ← PC
PC ← PC + 2 + REL

Description : Branches to the specified address after saving the PC contents to the stack. Use an 11-bit value relative to PC (–1024 to +1023) to specify the branch address.

- ❋ This instruction cannot be used in branching across bank boundaries.
- ❋ Do not place this instruction at bank boundaries.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BSR rr	$(11111b_{10}\ b_9\ b_8)_2$, $(b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)_2$ ❋ b_{10} to b_0 means "b10 to b0 of rr."	2	7

Note : Any value from –1023 to 1024 (11-bit length) can be set to rr.

Description example:

```
BSR     LABEL     ; Branches to LABEL
```

Function : Conditional branch

Operation data length: 16 bits or 8 bits

Operation : Relative branch to the specified address when A (bit n) = "1" or M (bit n) = "1" (n = 0 to 15. Only 1 bit can be specified).

Description : Branches to the specified address if the contents of the specified bit of A or a memory is "1." Use an 8-bit value relative to PC (–128 to +127) to specify the branch address. The bit position to be tested is specified by the bit number.

- When m = "0" : Any 1 bit between b0 to b15 can be specified.
  When m = "1" : Any 1 bit between b0 to b7 can be specified.
- B cannot be used in this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	BSS n, A, rr	01_{16}, n+80_{16}, rr	3	7
DIR	BSS n, dd, rr	71_{16}, n+80_{16}, dd, rr	4	11
ABS	BSS n, mmll, rr	71_{16}, n+$C0_{16}$, ll, mm, rr	5	10

Note : Any value from 0 to 15 can be set to n.

Description example:

```
CLM
BSS     8, A, LABEL1        ; Branches to LABEL1 if b8 of A is "1."
BSS     15, MEM16, LABEL2   ; Branches to LABEL2 if b15 of MEM16 is "1."
SEM
BSS     7, A, LABEL3        ; Branches to LABEL3 if b7 of A is "1."
BSS     7, MEM8, LABEL4     ; Branches to LABEL4 if b7 of MEM8 is "1."
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when V = “0.”

Description : Branches to the specified address if the contents of flag V is “0.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BVC rr	50_{16}, rr	2	6

Description example:

```
BVC     LABEL     ; Branches to LABEL if V = “0.”
```

Function : Conditional branch

Operation data length: –

Operation : Relative branch to the specified address when V = “1.”

Description : Branches to the specified address if the contents of flag V are “1.” Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
REL	BVS rr	70_{16}, rr	2	6

Description example:

```
BVS        LABEL        ; Branches to LABEL if V = “1.”
```

Function : Comparison & Conditional branch

Operation data length: 16 bits or 8 bits

Operation : Relative branch to the specified address when Acc = IMM or M = IMM.

Description : Branches to the specified address if the contents of Acc or a memory are equal to the immediate value. Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- When m = "0" : This instruction operates in 16-bit length.
  When m = "1" : This instruction operates in 8-bit length.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow is occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	CBEQ A, #imm, rr	$A6_{16}$, imm, rr	3	6
A	CBEQ B, #imm, rr	81_{16}, $A6_{16}$, imm, rr	4	7
DIR	CBEQ dd, #imm, rr	41_{16}, $6A_{16}$, dd, imm, rr	5	9

Note : When flag m = "0," the byte number increases by 1.

Description example:

```
CLM
CBEQ.W    A, #IMM16, LABEL1        ; Branches to LABEL1 if A = IMM16.
CBEQ.W    MEM16, #IMM16, LABEL2    ; Branches to LABEL2 if MEM16 = IMM16.
SEM
CBEQ.B    B, #IMM8, LABEL3         ; Branches to LABEL3 if BL = IMM8.
```

Function : Comparison & Conditional branch

Operation data length: 8 bits

Operation : Relative branch to the specified address when Acc_L = IMM8 or M8 = IMM8.

Description : Branches to the specified address if the contents of Acc_L or a memory are equal to the immediate value when they are compared in 8-bit length. Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	CBEQB A, #imm, rr	$A2_{16}$, imm, rr	3	6
A	CBEQB B, #imm, rr	81_{16}, $A2_{16}$, imm, rr	4	7
DIR	CBEQB dd, #imm, rr	62_{16}, dd, imm, rr	4	8

Description example:

```
CBEQB   A, #IMM8, LABEL1    ; Branches to LABEL1 if AL = IMM8.
CBEQB   MEM8, #IMM8, LABEL2 ; Branches to LABEL2 if MEM8 = IMM8.
```

Function : Comparison & Conditional branch

Operation data length: 16 bits or 8 bits

Operation : Relative branch to the specified address when Acc ≠ IMM or M ≠ IMM.

Description : Branches to the specified address if the contents of Acc or a memory are not equal to the immediate value. Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- When m = "0" : This instruction operates in 16-bit length.
  When m = "1" : This instruction operates in 8-bit length.
  ❋ In this case, the contents of Acc_H do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	CBNE A, #imm, rr	$B6_{16}$, imm, rr	3	6
A	CBNE B, #imm, rr	81_{16}, $B6_{16}$, imm, rr	4	7
DIR	CBNE dd, #imm, rr	41_{16}, $7A_{16}$, dd, imm, rr	5	9

Note : When flag m = "0," the byte number increases by 1.

Description example:

```
CLM
CBNE.W   A, #IMM16, LABEL1      ; Branches to LABEL1 if A ≠ IMM16.
CBNE.W   MEM16, #IMM16, LABEL2  ; Branches to LABEL2 if MEM16 ≠ IMM16.
```

Function : Comparison & Conditional branch

Operation data length: 8 bits

Operation : Relative branch to the specified address when $Acc_L \neq$ IMM8 or M8 ≠ IMM8.

Description : Branches to the specified address if the contents of Acc_L or a memory are equal to the immediate value when they are compared in 8-bit length. Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	CBNEB A, #imm, rr	$B2_{16}$, imm, rr	3	6
A	CBNEB B, #imm, rr	81_{16}, $B2_{16}$, imm, rr	4	7
DIR	CBNEB dd, #imm, rr	72_{16}, dd, imm, rr	4	8

Description example:

```
CBNEB   A, #IMM8, LABEL1     ; Branches to LABEL1 if AL ≠ IMM8.
CBNEB   MEM8, #IMM8, LABEL2  ; Branches to LABEL2 if MEM8 ≠ IMM8.
```

Function : Flag manipulation

Operation data length: –

Operation : C ← 0

Description : Clears the contents of flag C to “0.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	0

C : Cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	CLC	14_{16}	1	1

Description example:

```
CLC                    ; C ← 0
```

Function : Flag manipulation

Operation data length: –

Operation : I ← 0

Description : Clears the contents of flag I to “0.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	0	—	—

I : Cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	CLI	15_{16}	1	3

Description example:

```
CLI                    ; I ← 0
```

Function : Flag manipulation

Operation data length: –

Operation : m ← 0

Description : Clears the contents of flag m to “0.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	0	—	—	—	—	—

m : Cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	CLM	45_{16}	1	3

Description example:

```
CLM                    ; m ← 0
```

Function : Flag manipulation

Operation data length: –

Operation : PS_L (bit n) ← 0 (n = 0 to 7. Multiple bits can be specified.)

Description : Clears the specified flags (multiple flags can be specified) of PS_L to "0." The flag positions (bits' positions in PS_L) to be specified are indicated by a bit pattern of an 8-bit immediate value, in which the bits set to "1" are the subject bits to be specified.

- This instruction is unaffected by flag m.

PS_L

b7	b6	b5	b4	b3	b2	b1	b0
N	V	m	x	D	I	Z	C

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	m	x	D	I	Z	C

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	CLP #imm	98_{16}, imm	2	4

Description example:

```
CLP     #IMM8     ; The specified bits of PSL ← 0
```

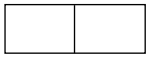
Function : Clear

Operation data length: 16 bits or 8 bits

Operation : Acc ← 0

When m = “0”

Acc



← 0000_{16}

When m = “1”

Acc_L

← 00_{16}

❋ In this case, the contents of Acc_H do not change.

Description : Clears the contents of Acc to “0.”

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	1	—

N : Always cleared to “0” because MSB of the operation result is “0.”

Z : Always set to “1” because the operation result is “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	CLR A	54_{16}	1	1
A	CLR B	81_{16}, 54_{16}	2	2

Description example:

```
CLM
CLR     A       ; A ← 0000₁₆
CLR     B       ; B ← 0000₁₆
SEM
CLR     A       ; AL ← 00₁₆
CLR     B       ; BL ← 00₁₆
```

Function : Clear

Operation data length: 8 bits

Operation : $Acc_L \leftarrow 00_{16}$

Acc_L
□ $\leftarrow 00_{16}$

Description : Clears the contents of Acc_L to "00_{16}."
- The contents of Acc_H do not change.
- This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	1	—

N : Always cleared to "0" because MSB of the operation result is "0."
Z : Always set to "1" because the operation result is "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	CLRB A	44_{16}	1	1
A	CLRB B	81_{16}, 44_{16}	2	2

Description example:

```
CLRB    A    ; AL ← 00₁₆
CLRB    B    ; BL ← 00₁₆
```

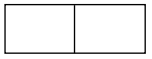
Function : Clear

Operation data length: 16 bits or 8 bits

Operation : M ← 0

When m = “0”

M16



← 0000_{16}

When m = “1”

M8

← 00_{16}

Description : Clears the contents of a memory to “0.”

Status flags

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	CLRM dd	$D2_{16}$, dd	2	5
ABS	CLRM mmll	$D7_{16}$, ll, mm	3	5

Description example:

```
CLM
CLRM    MEM16    ; MEM16 ← 0000₁₆
SEM
CLRM    MEM8     ; MEM8 ← 00₁₆
```

Function : Clear

Operation data length: 8 bits

Operation : $M8 \leftarrow 00_{16}$

M8
□ $\leftarrow 00_{16}$

Description : Clears the contents of a memory to “0” in 8-bit length.
● This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	○	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	CLRMB dd	$C2_{16}$, dd	2	5
ABS	CLRMB mmll	$C7_{16}$, ll, mm	3	5

Description example:

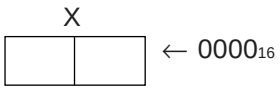
```
CLRMB    MEM8    ; MEM8 ← 00₁₆
```

Function : Clear

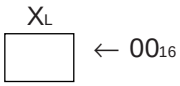
Operation data length: 16 bits or 8 bits

Operation : $X \leftarrow 0$

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Clears the contents of X to "0."

● This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	1	—

N : Always cleared to "0" because MSB of the operation result is "0."

Z : Always set to "1" because the operation result is "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	CLRX	$E4_{16}$	1	1

Description example:

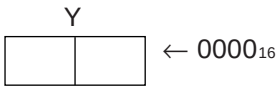
```
CLP   x
CLRX                ; X ← 0000₁₆
SEP   x
CLRX                ; XL ← 00₁₆
```

Function : Clear

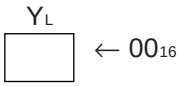
Operation data length: 16 bits or 8 bits

Operation : $Y \leftarrow 0$

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Clears the contents of Y to "0."

● This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	1	—

N : Always cleared to "0" because MSB of the operation result is "0."

Z : Always set to "1" because the operation result is "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	CLRY	$F4_{16}$	1	1

Description example:

```
CLP     x
CLRY                ; Y ← 0000₁₆
SEP     x
CLRY                ; YL ← 00₁₆
```

Function : Flag manipulation

Operation data length: –

Operation : V ← 0

Description : Clears the contents of flag V to “0.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	0	—	—	—	—	—	—

V : Cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	CLV	65_{16}	1	1

Description example:

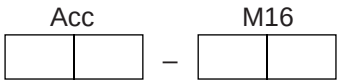
```
CLV                    ; V ← 0
```

Function : Comparison

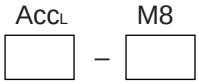
Operation data length: 16 bits or 8 bits

Operation : Acc – M

When m = “0”



When m = “1”



Description : Subtracts the contents of a memory from the contents of Acc. The result is not stored anywhere.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	CMP A, #imm	46_{16}, imm (81_{16}, 46_{16}, imm)	2 (3)	1 (2)
DIR	CMP A, dd	$4A_{16}$, dd (81_{16}, $4A_{16}$, dd)	2 (3)	3 (4)
DIR, X	CMP A, dd, X	$4B_{16}$, dd (81_{16}, $4B_{16}$, dd)	2 (3)	4 (5)
(DIR)	CMP A, (dd)	11_{16}, 40_{16}, dd (91_{16}, 40_{16}, dd)	3 (3)	6 (6)
(DIR, X)	CMP A, (dd, X)	11_{16}, 41_{16}, dd (91_{16}, 41_{16}, dd)	3 (3)	7 (7)
(DIR), Y	CMP A, (dd), Y	11_{16}, 48_{16}, dd (91_{16}, 48_{16}, dd)	3 (3)	7 (7)
L(DIR)	CMP A, L(dd)	11_{16}, 42_{16}, dd (91_{16}, 42_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	CMP A, L(dd), Y	11_{16}, 49_{16}, dd (91_{16}, 49_{16}, dd)	3 (3)	9 (9)
SR	CMP A, nn, S	11_{16}, 43_{16}, nn (91_{16}, 43_{16}, nn)	3 (3)	5 (5)
(SR), Y	CMP A, (nn, S), Y	11_{16}, 44_{16}, nn (91_{16}, 44_{16}, nn)	3 (3)	8 (8)
ABS	CMP A, mmll	$4E_{16}$, ll, mm (81_{16}, $4E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	CMP A, mmll, X	$4F_{16}$, ll, mm (81_{16}, $4F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	CMP A, mmll, Y	11_{16}, 46_{16}, ll, mm (91_{16}, 46_{16}, ll, mm)	4 (4)	5 (5)
ABL	CMP A, hhmmll	11_{16}, $4C_{16}$, ll, mm, hh (91_{16}, $4C_{16}$, ll, mm, hh)	5 (5)	5 (5)
ABL, X	CMP A, hhmmll, X	11_{16}, $4D_{16}$, ll, mm, hh (91_{16}, $4D_{16}$, ll, mm, hh)	5 (5)	6 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace "A" with "B" in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

Description example:

```
CLM
CMP.W    A, #IMM16          ; A – IMM16
CMP      B, MEM16           ; B – MEM16
SEM
CMP.B    A, #IMM8           ; AL – IMM8
CMP      B, MEM8            ; BL – MEM8
```

Function : Comparison

Operation data length: 8 bits

Operation : Acc_L – IMM8

$$\frac{Acc_L}{\square} - IMM8$$

Description : Subtracts the immediate value from the contents of Acc_L in 8-bit length. The result is not stored anywhere.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	CMPB A, #imm	38_{16}, imm	2	1
IMM	CMPB B, #imm	81_{16}, 38_{16}, imm	3	2

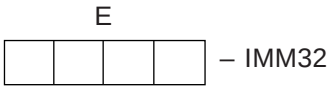
Description example:

```
CMPB    A, #IMM8        ; AL – IMM8
CMPB    B, #IMM8        ; BL – IMM8
```

Function : Comparison

Operation data length: 32 bits

Operation : E – IMM32



Description : Subtracts the immediate value from the contents of E in 32-bit length. The result is not stored anywhere.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	CMPD E, #imm	$3C_{16}$, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	5	3
DIR	CMPD E, dd	BA_{16}, dd	2	6
DIR, X	CMPD E, dd, X	BB_{16}, dd	2	7
(DIR)	CMPD E, (dd)	11_{16}, $B0_{16}$, dd	3	9
(DIR, X)	CMPD E, (dd, X)	11_{16}, $B1_{16}$, dd	3	10
(DIR), Y	CMPD E, (dd), Y	11_{16}, $B8_{16}$, dd	3	10
L(DIR)	CMPD E, L(dd)	11_{16}, $B2_{16}$, dd	3	11
L(DIR), Y	CMPD E, L(dd), Y	11_{16}, $B9_{16}$, dd	3	12
SR	CMPD E, nn, S	11_{16}, $B3_{16}$, nn	3	8
(SR), Y	CMPD E, (nn, S), Y	11_{16}, $B4_{16}$, nn	3	11
ABS	CMPD E, mmll	BE_{16}, ll, mm	3	6
ABS, X	CMPD E, mmll, X	BF_{16}, ll, mm	3	7
ABS, Y	CMPD E, mmll, Y	11_{16}, $B6_{16}$, ll, mm	4	8
ABL	CMPD E, hhmmll	11_{16}, BC_{16}, ll, mm, hh	5	8
ABL, X	CMPD E, hhmmll, X	11_{16}, BD_{16}, ll, mm, hh	5	9

Description example:

```
CMPD    E, #IMM32        ; E – IMM32
```

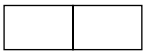
Function : Comparison

Operation data length: 16 bits or 8 bits

Operation : M – IMM

When m = “0”

M16

 – IMM16

When m = “1”

M8

 – IMM8

Description : Subtracts the immediate value from the contents of a memory. The result is not stored anywhere.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the result of the operation is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	CMPM dd, #imm	51_{16}, 23_{16}, dd, imm	4	5
ABS	CMPM mmll, #imm	51_{16}, 27_{16}, ll, mm, imm	5	5

Note : When flag m = “0.” the byte number increases by 1.

Description example:

```
CLM
CMPM.W    MEM16, #IMM16        ; MEM16 – IMM16
SEM
CMPM.B    MEM8, #IMM8          ; MEM8 – IMM8
```

Function : Comparison

Operation data length: 8 bits

Operation : M8 – IMM8

M8
□ – IMM8

Description : Subtracts the immediate value from the contents of a memory in 8-bit length. The result is not stored anywhere.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	CMPMB dd, #imm	51_{16}, 22_{16}, dd, imm	4	5
ABS	CMPMB mmll, #imm	51_{16}, 26_{16}, ll, mm, imm	5	5

Description example:

```
CMPMB    MEM8, #IMM8          ; MEM8 – IMM8
```

Function : Comparison

Operation data length: 32 bits

Operation : M32 – IMM32

M32

 – IMM32

Description : Subtracts the immediate value from the contents of a memory in 32-bit length. The result is not stored anywhere.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	CMPMD dd, #imm	51_{16}, $A3_{16}$, dd, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	7	7
ABS	CMPMD mmll, #imm	51_{16}, $A7_{16}$, ll, mm, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	8	7

Description example:

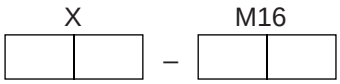
```
CMPMD    MEM32, #IMM32          ; MEM32 – IMM32
```

Function : Comparison

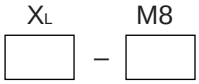
Operation data length: 16 bits or 8 bits

Operation : X – M

When x = "0"



When x = "1"



Description : Subtracts the contents of a memory from the contents of X. The result is not stored anywhere.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag x is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	CPX #imm	$E6_{16}$, imm	2	1
DIR	CPX dd	22_{16}, dd	2	3
ABS	CPX mmll	41_{16}, $2E_{16}$, ll, mm	4	4

Note : In the immediate addressing mode with flag x = "0," the byte number incleases by 1.

Description example:

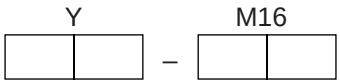
```
CLP     x
CPX.W   #IMM16      ; X – IMM16
CPX     MEM16       ; X – MEM16
SEP     x
CPX.B   #IMM8       ; XL – IMM8
CPX     MEM8        ; XL – MEM8
```

Function : Comparison

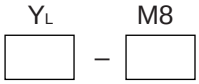
Operation data length: 16 bits or 8 bits

Operation : Y – M

When x = "0"



When x = "1"



Description : Subtracts the contents of a memory from the contents of Y. The result is not stored anywhere.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag x is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	CPY #imm	$F6_{16}$, imm	2	1
DIR	CPY dd	32_{16}, dd	2	3
ABS	CPY mmll	41_{16}, $3E_{16}$, ll, mm	4	4

Note : In the immediate addressing mode with flag x = "0," the byte number incleases by 1.

Description example:

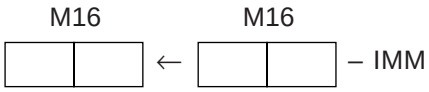
```
CLP     x
CPY.W   #IMM16      ; Y – IMM16
CPY     MEM16       ; Y – MEM16
SEP     x
CPY.B   #IMM8       ; YL – IMM8
CPY     MEM8        ; YL – MEM8
```

Function : Decrement & Conditional branch

Operation data length: 16 bits or 8 bits

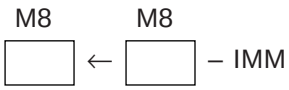
Operation : M ← M – IMM (IMM = 0 to 31)

When m = "0"



- When M16 (result of operation) = 0, executes the next instruction.
- When M16 (result of operation) ≠ 0, branches to the specified address.

When m = "1"



- When M8 (result of operation) = 0, executes the next instruction.
- When M8 (result of operation) ≠ 0, branches to the specified address.

Description : Subtracts the immediate value (0 to 31) from the contents of a memory, and stores the result to the memory. In this time, branches to the specified address, if the operation result is not "0." Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	DEBNE dd, #imm, rr	$C1_{16}$, imm+$A0_{16}$, dd, rr	4	12
ABS	DEBNE mmll, #imm, rr	$D1_{16}$, imm+$E0_{16}$, ll, mm, rr	5	11

Note : Any value from 0 to 31 can be set to imm.

Description example:

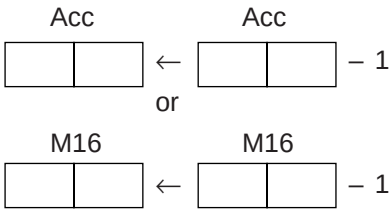
```
CLM
DEBNE    MEM16, #IMM, LABEL1   ; Branches to LABEL1, if the result of MEM16 – IMM(0 to 31) is not 0.
SEM
DEBNE    MEM8, #IMM, LABEL2    ; Branches to LABEL2, if the result of MEM8 – IMM(0 to 31) is not 0.
```

Function : Decrement

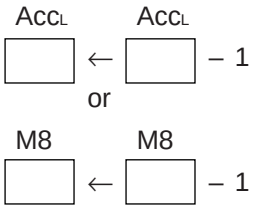
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc – 1 or M ← M – 1

When m = "0"



When m = "1"



❋ In this case, the contents of Acc_H do not change.

Description : Decrements 1 from the contents of Acc or the contents of a memory.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	DEC A	$B3_{16}$	1	1
A	DEC B	81_{16}, $B3_{16}$	2	2
DIR	DEC dd	92_{16}, dd	2	6
DIR, X	DEC dd, X	41_{16}, $9B_{16}$, dd	3	8
ABS	DEC mmll	97_{16}, ll, mm	3	6
ABS, X	DEC mmll, X	41_{16}, $9F_{16}$, ll, mm	4	8

Description example:

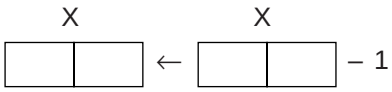
```
CLM
DEC     A          ; A ← A – 1
SEM
DEC     A          ; AL ← AL – 1
```

Function : Decrement

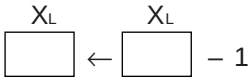
Operation data length: 16 bits or 8 bits

Operation : $X \leftarrow X - 1$

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Decrements 1 from the contents of X.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	DEX	$E3_{16}$	1	1

Description example:

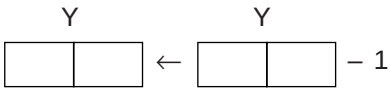
```
CLP   x
DEX                    ; X ← X – 1
SEP   x
DEX                    ; XL ← XL – 1
```

Function : Decrement

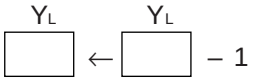
Operation data length: 16 bits or 8 bits

Operation : $Y \leftarrow Y - 1$

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Decrements 1 from the contents of Y.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	DEY	$F3_{16}$	1	1

Description example:

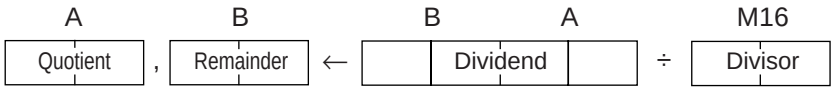
```
CLP   x
DEY                    ; Y ← Y – 1
SEP   x
DEY                    ; YL ← YL – 1
```

Function : Division (Unsigned)

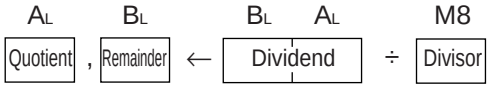
Operation data length: 16 bits or 8 bits

Operation : A (quotient), B (remainder) ← (B, A) ÷ M

When m = "0"



When m = "1"



❋ In this case, the contents of A_H and B_H do not change.

Description : Divides the data whose high-order bits consist of the contents of accumulator B and low-order bits consist of the contents of accumulator A by the memory's contents. Stores the quotient to accumulator A, and stores the remainder to accumulator B.

- If an overflow occurs as an operation result, flag V is set to "1" and the contents of accumulators A and B become undefined.
- When the divisor is "0," the zero divide interrupt is generated. In that case, the contents of accumulators A and B are not changed.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	I	Z	C

N : Set to "1" if the quotient (A as the operation result)'s MSB is "1." Unaffected when an overflow occurs or the divisor is "0." Otherwise, cleared to "0."

V : Set to "1" when an overflow occurs. Unaffected when the divisor is "0." Otherwise, cleared to "0."

I : Set to "1" when the divisor is "0." Otherwise, unaffected.

Z : Set to "1" when the quotient (A as the operation result) is "0." Unaffected when an overflow occurs or the divisor is "0." Otherwise, cleared to "0."

C : Set to "1" when an overflow occurs. Unaffected when the divisor is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	DIV #imm	31_{16}, $E7_{16}$, imm	3	15
DIR	DIV dd	21_{16}, EA_{16}, dd	3	16
DIR, X	DIV dd, X	21_{16}, EB_{16}, dd	3	17
(DIR)	DIV (dd)	21_{16}, $E0_{16}$, dd	3	18
(DIR, X)	DIV (dd, X)	21_{16}, $E1_{16}$, dd	3	19
(DIR), Y	DIV (dd), Y	21_{16}, $E8_{16}$, dd	3	19
L(DIR)	DIV L(dd)	21_{16}, $E2_{16}$, dd	3	20
L(DIR), Y	DIV L(dd), Y	21_{16}, $E9_{16}$, dd	3	21
SR	DIV nn, S	21_{16}, $E3_{16}$, nn	3	17
(SR), Y	DIV (nn, S), Y	21_{16}, $E4_{16}$, nn	3	20
ABS	DIV mmll	21_{16}, EE_{16}, ll, mm	4	16
ABS, X	DIV mmll, X	21_{16}, EF_{16}, ll, mm	4	17
ABS, Y	DIV mmll, Y	21_{16}, $E6_{16}$, ll, mm	4	17
ABL	DIV hhmmll	21_{16}, EC_{16}, ll, mm, hh	5	17
ABL, X	DIV hhmmll, X	21_{16}, ED_{16}, ll, mm, hh	5	18

Notes 1: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

2: The cycle number in this table applies to the case of 16-bit ÷ 8-bit operation. In the case of 32-bit ÷ 16-bit operation, the cycle number increases by 8.

3: The cycle number in this table and Note 2 is the number when the operation is completed normally (in other words, when no interrupt has been generated). If a zero divide interrupt is generated, the cycle number is 16 cycles regardless of the operation's data length.

Description example:

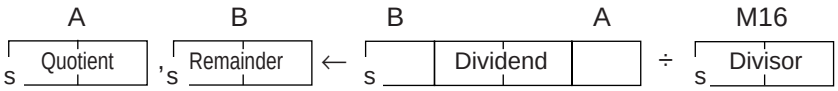
```
CLM
DIV     MEM16     ; A, B ← (B, A) / MEM16
DIV.W   #IMM16    ; A, B ← (B, A) / IMM16
SEM
DIV     MEM8      ; AL, BL ← (BL, AL) / MEM8
DIV.B   #IMM8     ; AL, BL ← (BL, AL) / IMM8
```

Function : Division (Signed)

Operation data length: 16 bits or 8 bits

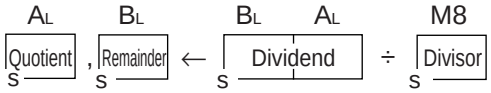
Operation : A (quotient), B (remainder) ← (B, A) ÷ M

When m = "0"



❋ "s" represents MSB of data.

When m = "1"



❋ "s" represents MSB of data.
❋ In this case, the contents of A_H and B_H do not change.

Description : Divides the signed data whose high-order bits consist of the contents of accumulator B and low-order bits consist of the contents of accumulator A by the memory's contents (signed). Stores the signed quotient to accumulator A, and stores the signed remainder to accumulator B.

- The sign of remainder becomes same as that of dividend.
- If an overflow occurs as an operation result (the quotient exceeds the range –32767 to +32767 when flag m is "0," or –127 to +127 when flag m is "1"), the operation finishes halfway and flag V is set to "1." In that case, the contents of accumulators A and B become undefined.
- When the divisor is "0," the zero divide interrupt is generated. In that case, the contents of accumulators A and B are not changed.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	I	Z	C

N : Set to "1" if the quotient (A as the operation result)'s MSB is "1." Unaffected when an overflow occurs or the divisor is "0." Otherwise, cleared to "0."

V : Set to "1" when an overflow occurs. Unaffected when the divisor is "0." Otherwise, cleared to "0."

I : Set to "1" when the divisor is "0." Otherwise, unaffected.

Z : Set to "1" when the quotient (A as the operation result) is "0." Unaffected when an overflow occurs or the divisor is "0." Otherwise, cleared to "0."

C : Set to "1" when an overflow occurs. Unaffected when the divisor is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	DIVS #imm	31_{16}, $F7_{16}$, imm	3	22
DIR	DIVS dd	21_{16}, FA_{16}, dd	3	23
DIR, X	DIVS dd, X	21_{16}, FB_{16}, dd	3	24
(DIR)	DIVS (dd)	21_{16}, $F0_{16}$, dd	3	25
(DIR, X)	DIVS (dd, X)	21_{16}, $F1_{16}$, dd	3	26
(DIR), Y	DIVS (dd), Y	21_{16}, $F8_{16}$, dd	3	26
L(DIR)	DIVS L(dd)	21_{16}, $F2_{16}$, dd	3	27
L(DIR), Y	DIVS L(dd), Y	21_{16}, $F9_{16}$, dd	3	28
SR	DIVS nn, S	21_{16}, $F3_{16}$, nn	3	24
(SR), Y	DIVS (nn, S), Y	21_{16}, $F4_{16}$, nn	3	27
ABS	DIVS mmll	21_{16}, FE_{16}, ll, mm	4	23
ABS, X	DIVS mmll, X	21_{16}, FF_{16}, ll, mm	4	24
ABS, Y	DIVS mmll, Y	21_{16}, $F6_{16}$, ll, mm	4	24
ABL	DIVS hhmmll	21_{16}, FC_{16}, ll, mm, hh	5	24
ABL, X	DIVS hhmmll, X	21_{16}, FD_{16}, ll, mm, hh	5	25

Notes 1: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

2: The cycle number in this table applies to the case of 16-bit ÷ 8-bit operation. In the case of 32-bit ÷ 16-bit operation, the cycle number increases by 8.

3: The cycle number in this table and Note 2 is the number when the operation is completed normally (in other words, when no interrupt has been generated). If a zero divide interrupt is generated, the cycle number is 16 cycles regardless of the operation's data length.

Description example:

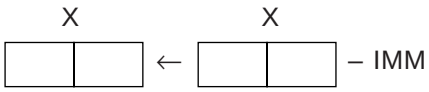
```
CLM
DIVS      MEM16      ; A, B ← (B, A) / MEM16
SEM
DIVS.B    #IMM8      ; AL, BL ← (BL, AL) / IMM8
```

Function : Decrement & Conditional branch

Operation data length: 16 bits or 8 bits

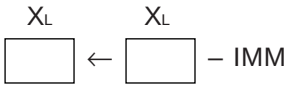
Operation : X ← X – IMM (IMM = 0 to 31)

When x = "0"



- When X (result of operation) = 0, executes the next instruction.
- When X (result of operation) ≠ 0, branches to the specified address.

When x = "1"



- When X_L (result of operation) = 0, executes the next instruction.
- When X_L (result of operation) ≠ 0, branches to the specified address.
- ❋ In this case, the contents of X_H do not change.

Description : Subtracts the immediate value (0 to 31) from the contents of X, and stores the result to the X. In this time, branches to the specified address, if the operation result is not "0." Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	DXBNE #imm, rr	01_{16}, imm+$C0_{16}$, rr	3	7

Note : Any value from 0 to 31 can be set to imm.

Description example:

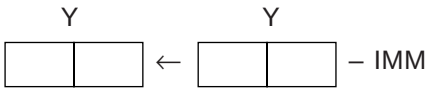
```
CLP     x
DXBNE   #IMM, LABEL1    ; Branches to LABEL1, if the result of X – IMM(0 to 31) is not 0.
SEP     x
DXBNE   #IMM, LABEL2    ; Branches to LABEL2, if the result of XL – IMM(0 to 31) is not 0.
```

Function : Decrement & Conditional branch

Operation data length: 16 bits or 8 bits

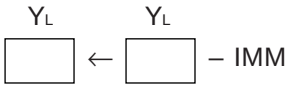
Operation : $Y \leftarrow Y - IMM$ (IMM = 0 to 31)

When x = "0"



- When Y (result of operation) = 0, executes the next instruction.
- When Y (result of operation) ≠ 0, branches to the specified address.

When x = "1"



- When Y_L (result of operation) = 0, executes the next instruction.
- When Y_L (result of operation) ≠ 0, branches to the specified address.
- ❋ In this case, the contents of Y_H do not change.

Description : Subtracts the immediate value (0 to 31) from the contents of Y, and stores the result to the Y. In this time, branches to the specified address, if the result of the operation is not "0." Use an 8-bit value relative to PC (–128 to +127) to specify the branch address.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	DYBNE #imm, rr	01_{16}, imm+$E0_{16}$, rr	3	7

Note : Any value from 0 to 31 can be set to imm.

Description example:

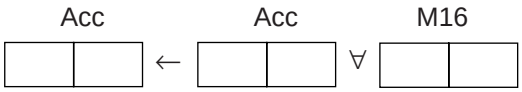
```
CLP     x
DYBNE   #IMM, LABEL1    ; Branches to LABEL1, if the result of Y – IMM(0 to 31) is not 0.
SEP     x
DYBNE   #IMM, LABEL2    ; Branches to LABEL2, if the result of YL – IMM(0 to 31) is not 0.
```

Function : Logical exclusive OR

Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc ∀ M

When m = "0"



When m = "1"

✵ In this case, the contents of Acc_H do not change.

Description : Performs the logical exclusive OR between the contents of Acc and the contents of a memory by each bit, and stores the result in Acc.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	EOR A, #imm	76_{16}, imm (81_{16}, 76_{16}, imm)	2 (3)	1 (2)
DIR	EOR A, dd	$7A_{16}$, dd (81_{16}, $7A_{16}$, dd)	2 (3)	3 (4)
DIR, X	EOR A, dd, X	$7B_{16}$, dd (81_{16}, $7B_{16}$, dd)	2 (3)	4 (5)
(DIR)	EOR A, (dd)	11_{16}, 70_{16}, dd (91_{16}, 70_{16}, dd)	3 (3)	6 (6)
(DIR, X)	EOR A, (dd, X)	11_{16}, 71_{16}, dd (91_{16}, 71_{16}, dd)	3 (3)	7 (7)
(DIR), Y	EOR A, (dd), Y	11_{16}, 78_{16}, dd (91_{16}, 78_{16}, dd)	3 (3)	7 (7)
L(DIR)	EOR A, L(dd)	11_{16}, 72_{16}, dd (91_{16}, 72_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	EOR A, L(dd), Y	11_{16}, 79_{16}, dd (91_{16}, 79_{16}, dd)	3 (3)	9 (9)
SR	EOR A, nn, S	11_{16}, 73_{16}, nn (91_{16}, 73_{16}, nn)	3 (3)	5 (5)
(SR), Y	EOR A, (nn, S), Y	11_{16}, 74_{16}, nn (91_{16}, 74_{16}, nn)	3 (3)	8 (8)
ABS	EOR A, mmll	$7E_{16}$, ll, mm (81_{16}, $7E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	EOR A, mmll, X	$7F_{16}$, ll, mm (81_{16}, $7F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	EOR A, mmll, Y	11_{16}, 76_{16}, ll, mm (91_{16}, 76_{16}, ll, mm)	4 (4)	5 (5)
ABL	EOR A, hhmmll	11_{16}, $7C_{16}$, ll, mm, hh (91_{16}, $7C_{16}$, ll, mm, hh)	5 (5)	5 (5)
ABL, X	EOR A, hhmmll, X	11_{16}, $7D_{16}$, ll, mm, hh (91_{16}, $7D_{16}$, ll, mm, hh)	5 (5)	6 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace “A” with “B” in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = “0.”

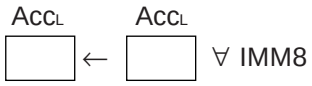
Description example:

```
CLM
EOR.W    A, #IMM16        ; A ← A ∀ IMM16
EOR      B, MEM16         ; B ← B ∀ MEM16
SEM
EOR.B    A, #IMM8         ; AL ← AL ∀ IMM8
EOR      B, MEM8          ; BL ← BL ∀ MEM8
```

Function : Logical exclusive OR

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L \forall IMM8$



Description : Performs the logical exclusive OR in 8-bit length between the contents of Acc_L and the contents of a memory by each bit, and stores the result in Acc_L.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	EORB A, #imm	33_{16}, imm	2	1
IMM	EORB B, #imm	81_{16}, 33_{16}, imm	3	2

Description example:

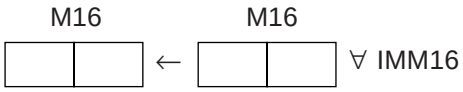
```
EORB    A, #IMM8        ; AL ← AL ∀ IMM8
EORB    B, #IMM8        ; BL ← BL ∀ IMM8
```

Function : Logical exclusive OR

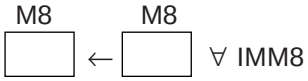
Operation data length: 16 bits or 8 bits

Operation : M ← M ∀ IMM

When m = "0"



When m = "1"



Description : Performs the logical exclusive OR between the contents of a memory and the immediate value, and stores the result in the memory.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	EORM dd, #imm	51_{16}, 73_{16}, dd, imm	4	7
ABS	EORM mmll, #imm	51_{16}, 77_{16}, ll, mm, imm	5	7

Note : When flag m = "0," the byte number increases by 1.

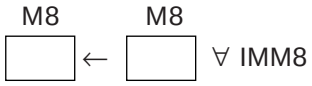
Description example:

```
CLM
EORM.W    MEM16, #IMM16        ; MEM16 ← MEM16 ∀ IMM16
SEM
EORM.B    MEM8, #IMM8          ; MEM8 ← MEM8 ∀ IMM8
```

Function : Logical exclusive OR

Operation data length: 8 bits

Operation : M8 ← M8 ∀ IMM8



Description : Performs the logical exclusive OR in 8-bit length between the contents of a memory and the immediate value, and stores the result in the memory.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	EORMB dd, #imm	51_{16}, 72_{16}, dd, imm	4	7
ABS	EORMB mmll, #imm	51_{16}, 76_{16}, ll, mm, imm	5	7

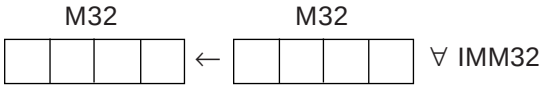
Description example:

```
EORMB    MEM8, #IMM8          ; MEM8 ← MEM8 ∀ IMM8
```

Function : Logical exclusive OR

Operation data length: 32 bits

Operation : M32 ← M32 ∀ IMM32



Description : Performs the logical exclusive OR in 32-bit length between the contents of a memory and the immediate value, and stores the result in the memory.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	EORMD dd, #imm	51_{16}, $F3_{16}$, dd, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	7	10
ABS	EORMD mmll, #imm	51_{16}, $F7_{16}$, ll, mm, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	8	10

Description example:

```
EORMD    MEM32, #IMM32        ; MEM32 ← MEM32 ∀ IMM32
```

Function : Extension sign

Operation data length: 16 bits

Operation : $Acc \leftarrow Acc_L$ (Extension sign)

When bit 7 of Acc_L = "0"

$Acc_H \leftarrow 00_{16}$

Acc_H	Acc_L		Acc_H	Acc_L
00_{16}	$0XXXXXXX_2$	←	?	$0XXXXXXX_2$

When bit 7 of Acc_L = "1"

$Acc_H \leftarrow FF_{16}$

Acc_H	Acc_L		Acc_H	Acc_L
FF_{16}	$1XXXXXXX_2$	←	?	$1XXXXXXX_2$

❋ The contents of Acc_H change regardless of flag m.

Description : This instruction is used to extend Acc_L to Acc with signs.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when bit 15 of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	EXTS A	35_{16}	1	1
A	EXTS B	81_{16}, 35_{16}	2	2

Description example:

```
EXTS    A       ; AH ← 0016 or FF16
EXTS    B       ; BH ← 0016 or FF16
```

Function : Extension sign

Operation data length: 32 bits

Operation : $E \leftarrow E_L$ (= A) (Extension sign)

When bit 15 of A = "0"

$E_H \leftarrow 0000_{16}$

E_H (= B)		E_L (= A)			E_H (= B)		E_L (= A)	
b15	b0	b15	b0		b15	b0	b15	b0
0000_{16}		$0X\cdots XX_2$		←	?		$0X\cdots XX_2$	

When bit 15 of A = "1"

$E_H \leftarrow FFFF_{16}$

E_H (= B)		E_L (= A)			E_H (= B)		E_L (= A)	
b15	b0	b15	b0		b15	b0	b15	b0
$FFFF_{16}$		$1X\cdots XX_2$		←	?		$1X\cdots XX_2$	

❋ The high-order 2 bytes change regardless of flag m.

Description : This instruction is used to extend E_L (= A) to E with signs.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
–	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	EXTSD E	31_{16}, $B0_{16}$	2	5

Description example:

```
EXTSD    E          ; E ← EL
                    ; (B ← 0000₁₆ or FFFF₁₆, A ← A)
```

Function : Extension zero

Operation data length: 16 bits

Operation : Acc ← Acc_L (Extension zero)

Acc_H	Acc_L		Acc_H	Acc_L
00_{16}		←	?	

✵ The contents of Acc_H change regardless of flag m.

Description : This instruction is used to extend Acc_L to Acc with 0s.
- This instruction is unaffected by flag m.
- The content of Acc_H always set to "00_{16}."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."
Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	EXTZ A	34_{16}	1	1
A	EXTZ B	81_{16}, 34_{16}	2	2

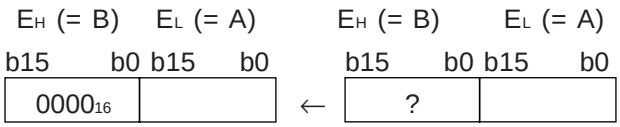
Description example:

```
EXTZ    A        ; A ← AL (AH ← 00₁₆ , AL ← AL)
EXTZ    B        ; B ← BL (BH ← 00₁₆ , BL ← BL)
```

Function : Extension zero

Operation data length: 32 bits

Operation : $E \leftarrow E_L$ (= A) (Extension zero)



❋ The high-order 2 bytes change regardless of flag m.

Description : This instruction is used to extend E_L (= A) to E with 0s.

- This instruction is unaffected by flag m.
- The high-order word; E_H (= B) becomes "0000_{16}."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	—

N : Always "0" because MSB of the operation result is "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	EXTZD E	31_{16}, $A0_{16}$	2	3

Description example:

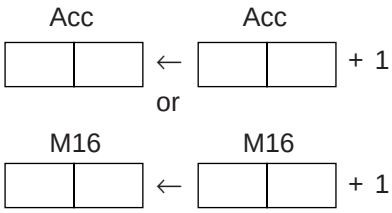
```
EXTZD    E                    ; E ← EL (B ← 0000₁₆, A ← A)
```

Function : Increment

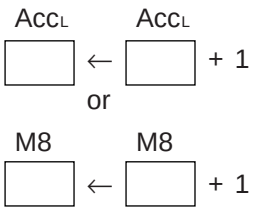
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc + 1 or M ← M + 1

When m = "0"



When m = "1"



❋ In this case, the contents of Acc_H do not change.

Description : Adds 1 to the contents of Acc or a memory.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	INC A	$A3_{16}$	1	1
A	INC B	81_{16}, $A3_{16}$	2	2
DIR	INC dd	82_{16}, dd	2	6
DIR, X	INC dd, X	41_{16}, $8B_{16}$, dd	3	8
ABS	INC mmll	87_{16}, ll, mm	3	6
ABS, X	INC mmll, X	41_{16}, $8F_{16}$, ll, mm	4	8

Description example:

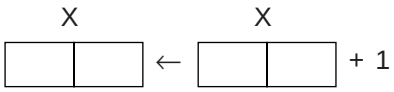
```
CLM
INC     A          ; A ← A + 1
INC     MEM16      ; MEM16 ← MEM16 + 1
SEM
INC     B          ; BL ← BL + 1
INC     MEM8       ; MEM8 ← MEM8 + 1
```

Function : Increment

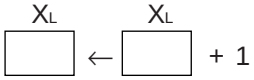
Operation data length: 16 bits or 8 bits

Operation : $X \leftarrow X + 1$

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Adds 1 to the contents of X.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	INX	$C3_{16}$	1	1

Description example:

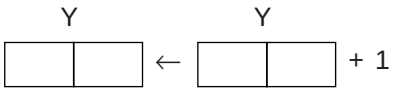
```
CLP   x
INX                    ; X ← X + 1
SEP   x
INX                    ; XL ← XL + 1
```

Function : Increment

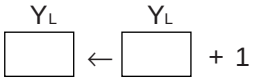
Operation data length: 16 bits or 8 bits

Operation : $Y \leftarrow Y + 1$

When x = "0"



When x = "1"



✵ In this case, the contents of Y_H do not change.

Description : Adds 1 to the contents of Y.

● This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	INY	$D3_{16}$	1	1

Description example:

```
CLP   x
INY                    ; Y ← Y + 1
SEP   x
INY                    ; YL ← YL + 1
```

Function : Jump always

Operation data length: –

Operation :
- JMP instruction

  PC ← Specified address

  PC ← mmll
- JMPL instruction

  PG, PC ← Specified address

  PC ← mmll

  PG ← hh

Description : Jumps to the specified address. Use a 16-bit (JMP) or 24-bit (JMPL) address to specify the destination jump address.

- If the last byte of the JMP instruction is placed at the highest address ($XXFFFF_{16}$) or the instruction is located across bank boundaries, the contents of PG are incremented by 1, causing control to jump to the specified address in the next bank.
- When using indirect addressing, the memory to be referenced is in the same program bank (the bank indicated by PG).

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
ABS	JMP mmll	$9C_{16}$, ll, mm	3	4
ABL	JMPL hhmmll	AC_{16}, ll, mm, hh	4	5
(ABS)	JMP (mmll)	31_{16}, $5C_{16}$, ll, mm	4	7
L(ABS)	JMPL L(mmll)	31_{16}, $5D_{16}$, ll, mm	4	9
(ABS, X)	JMP (mmll, X)	BC_{16}, ll, mm	3	7

Description example:

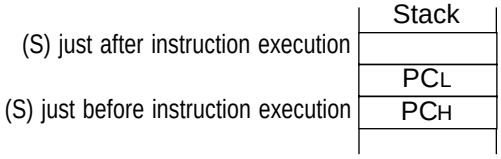
```
JMP     ADDR16      ; Jump to the address ADDR16
JMPL    ADDR24      ; Jump to the address ADDR24
```

Function : Subroutine call

Operation data length: –

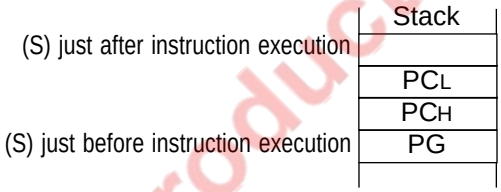
Operation : • JSR instruction

Stack ← PC
PC ← Specified address
PC ← PC + 3
M(S, S – 1) ← PC
S ← S – 2
PC ← mmll



• JSRL instruction

Stack ← PG, PC
PG, PC ← Specified address
PC ← PC + 4
M(S to S – 2) ← PG, PC
S ← S – 3
PC ← mmll
PG ← hh



Description : This instruction stores the contents of PG and PC to stack, and jumps to the specified address. Use a 16-bit (JSR) or 24-bit (JSRL) address to specify the destination jump address.

- If the last byte of the JSR instruction is placed at the highest address ($XXFFFF_{16}$) or the instruction is located across bank boundaries, the contents of PG are incremented by 1, causing control to jump to the specified address in the next bank.
- When using indirect addressing, the memory to be referenced is in the same program bank (the bank indicated by PG).

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
ABS	JSR mmll	$9D_{16}$, ll, mm	3	6
ABL	JSRL hhmmll	AD_{16}, ll, mm, hh	4	7
(ABS, X)	JSR (mmll, X)	BD_{16}, ll, mm	3	8

Description example:

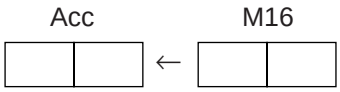
```
JSR     ADDR16      ; Jump to the address ADDR16
JSRL    ADDR24      ; Jump to the address ADDR24
```

Function : Load

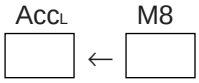
Operation data length: 16 bits or 8 bits

Operation : Acc ← M

When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Loads the contents of a memory into Acc.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDA A, #imm	16_{16}, imm (81_{16}, 16_{16}, imm)	2 (3)	1 (2)
DIR	LDA A, dd	$1A_{16}$, dd (81_{16}, $1A_{16}$, dd)	2 (3)	3 (4)
DIR, X	LDA A, dd, X	$1B_{16}$, dd (81_{16}, $1B_{16}$, dd)	2 (3)	4 (5)
(DIR)	LDA A, (dd)	11_{16}, 10_{16}, dd (91_{16}, 10_{16}, dd)	3 (3)	6 (6)
(DIR, X)	LDA A, (dd, X)	11_{16}, 11_{16}, dd (91_{16}, 11_{16}, dd)	3 (3)	7 (7)
(DIR), Y	LDA A, (dd), Y	18_{16}, dd (81_{16}, 18_{16}, dd)	2 (3)	6 (7)
L(DIR)	LDA A, L(dd)	11_{16}, 12_{16}, dd (91_{16}, 12_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	LDA A, L(dd), Y	19_{16}, dd (81_{16}, 19_{16}, dd)	2 (3)	8 (9)
SR	LDA A, nn, S	11_{16}, 13_{16}, nn (91_{16}, 13_{16}, nn)	3 (3)	5 (5)
(SR), Y	LDA A, (nn, S), Y	11_{16}, 14_{16}, nn (91_{16}, 14_{16}, nn)	3 (3)	8 (8)
ABS	LDA A, mmll	$1E_{16}$, ll, mm (81_{16}, $1E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	LDA A, mmll, X	$1F_{16}$, ll, mm (81_{16}, $1F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	LDA A, mmll, Y	11_{16}, 16_{16}, ll, mm (91_{16}, 16_{16}, ll, mm)	4 (4)	5 (5)
ABL	LDA A, hhmmll	$1C_{16}$, ll, mm, hh (81_{16}, $1C_{16}$, ll, mm, hh)	4 (5)	4 (5)
ABL, X	LDA A, hhmmll, X	$1D_{16}$, ll, mm, hh (81_{16}, $1D_{16}$, ll, mm, hh)	4 (5)	5 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace “A” with “B” in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = “0.”

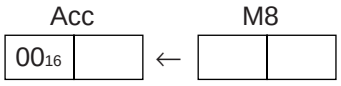
Description example:

```
CLM
LDA.W    A, #IMM16      ; A ← IMM16
LDA      B, MEM16       ; B ← MEM16
SEM
LDA.B    A, #IMM8       ; AL ← IMM8
LDA      B, MEM8        ; BL ← MEM8
```

Function : Load

Operation data length: 16 bits

Operation : Acc ← M8 (Extension zero)



Description : Transfers 8-bit data from memory to Acc after zero-extending it to 16 bits.

- This instruction is unaffected by flag m.
- The contents of Acc_H are always set to "00_{16}."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDAB A, #imm	28_{16}, imm (81_{16}, 28_{16}, imm)	2 (3)	1 (2)
DIR	LDAB A, dd	$0A_{16}$, dd (81_{16}, $0A_{16}$, dd)	2 (3)	3 (4)
DIR, X	LDAB A, dd, X	$0B_{16}$, dd (81_{16}, $0B_{16}$, dd)	2 (3)	4 (5)
(DIR)	LDAB A, (dd)	11_{16}, 00_{16}, dd (91_{16}, 00_{16}, dd)	3 (3)	6 (6)
(DIR, X)	LDAB A, (dd, X)	11_{16}, 01_{16}, dd (91_{16}, 01_{16}, dd)	3 (3)	7 (7)
(DIR), Y	LDAB A, (dd), Y	08_{16}, dd (81_{16}, 08_{16}, dd)	2 (3)	6 (7)
L(DIR)	LDAB A, L(dd)	11_{16}, 02_{16}, dd (91_{16}, 02_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	LDAB A, L(dd), Y	09_{16}, dd (81_{16}, 09_{16}, dd)	2 (3)	8 (9)
SR	LDAB A, nn, S	11_{16}, 03_{16}, nn (91_{16}, 03_{16}, nn)	3 (3)	5 (5)
(SR), Y	LDAB A, (nn, S), Y	11_{16}, 04_{16}, nn (91_{16}, 04_{16}, nn)	3 (3)	8 (8)
ABS	LDAB A, mmll	$0E_{16}$, ll, mm (81_{16}, $0E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	LDAB A, mmll, X	$0F_{16}$, ll, mm (81_{16}, $0F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	LDAB A, mmll, Y	11_{16}, 06_{16}, ll, mm (91_{16}, 06_{16}, ll, mm)	4 (4)	5 (5)
ABL	LDAB A, hhmmll	$0C_{16}$, ll, mm, hh (81_{16}, $0C_{16}$, ll, mm, hh)	4 (5)	4 (5)
ABL, X	LDAB A, hhmmll, X	$0D_{16}$, ll, mm, hh (81_{16}, $0D_{16}$, ll, mm, hh)	4 (5)	5 (6)

Note : This table applies when using accumulator A. When using accumulator B, replace "A" with "B" in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

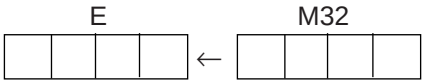
Description example:

```
LDAB    A, #IMM8        ; A ← IMM8 (AH ← 00₁₆, AL ← IMM8)
LDAB    B, MEM8         ; B ← MEM8 (BH ← 00₁₆, BL ← MEM8)
```

Function : Load

Operation data length: 32 bits

Operation : E ← M32



Description : Loads the 32-bit data of a memory to E.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDAD E, #imm	$2C_{16}$, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	5	3
DIR	LDAD E, dd	$8A_{16}$, dd	2	6
DIR, X	LDAD E, dd, X	$8B_{16}$, dd	2	7
(DIR)	LDAD E, (dd)	11_{16}, 80_{16}, dd	3	9
(DIR, X)	LDAD E, (dd, X)	11_{16}, 81_{16}, dd	3	10
(DIR), Y	LDAD E, (dd), Y	88_{16}, dd	2	9
L(DIR)	LDAD E, L(dd)	11_{16}, 82_{16}, dd	3	11
L(DIR), Y	LDAD E, L(dd), Y	89_{16}, dd	2	11
SR	LDAD E, nn, S	11_{16}, 83_{16}, nn	3	8
(SR), Y	LDAD E, (nn, S), Y	11_{16}, 84_{16}, nn	3	11
ABS	LDAD E, mmll	$8E_{16}$, ll, mm	3	6
ABS, X	LDAD E, mmll, X	$8F_{16}$, ll, mm	3	7
ABS, Y	LDAD E, mmll, Y	11_{16}, 86_{16}, ll, mm	4	8
ABL	LDAD E, hhmmll	$8C_{16}$, ll, mm, hh	4	7
ABL, X	LDAD E, hhmmll, X	$8D_{16}$, ll, mm, hh	4	8

Description example:

```
LDAD    E, #IMM32       ; E ← IMM32
                        ;  (B ← IMM32H, A ← IMM32L)
LDAD    E, MEM32        ; E ← MEM32
                        ;  (B ← IMM32H, A ← IMM32L)
```

Function : Load

Operation data length: 16 bits

Operation : DPR0 ← IMM16a (can be specified to multiple DPRs)
DPR1 ← IMM16b
DPR2 ← IMM16c
DPR3 ← IMM16d

DPR0 ← IMM16a

DPR1 ← IMM16b

DPR2 ← IMM16c

DPR3 ← IMM16d

Description : Transfers a 16-bit immediate value to DPR0 through DPR3.

- This instruction is unaffected by flag m.
- A value can be set to multiple DPRs by 1 instruction. If multiple DPRs are specified, transfers are performed in order of DPR0, DPR1, DPR2, and DPR3.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDD n, #imm	$B8_{16}$, $?0_{16}$, imm_L, imm_H	4	13
	LDD (n_1, …, n_i), #imm_1, …, #imm_i	$B8_{16}$, $?0_{16}$, imm_{L1}, imm_{H1} , …, imm_{Li}, imm_{Hi}	2 × i + 2	2 × i + 11

Notes 1: Any value from 0 to 3 can be set to n.
2: The second line of the syntax format sets values to multiple DPRs by 1 instruction.
3: The inside of parentheses (n1, ..., ni) specifies 0 to 3 (numbers representing DPRn).
4: i: Indicates DPRn specified (1 to 4).
5: ?: The bit corresponding to a specified DPRn is set to "1." The diagram below shows the relationship between bits and DPRn.

b7							b0
DPR3	DPR2	DPR1	DPR0	0	0	0	0

Description example:

```
LDD     0, #IMM16           ; DPR0 ← IMM16
LDD     (0, 3), #IMM16a, #IMM16b
                            ; DPR0 ← IMM16a
                            ; DPR3 ← IMM16b
```

Function : Load

Operation data length: 8 bits

Operation : DT ← IMM8

DT
[ ] ← IMM8

Description : Loads the immediate value to DT.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDT #imm	31_{16}, $4A_{16}$, imm	3	4

Description example:

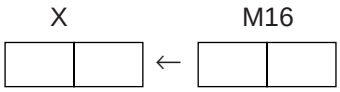
```
LDT     #IMM8     ; DT ← IMM8
```

Function : Load

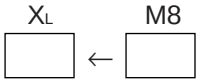
Operation data length: 16 bits or 8 bits

Operation : X ← M

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Loads the contents of a memory to X.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDX #imm	$C6_{16}$, imm	2	1
DIR	LDX dd	02_{16}, dd	2	3
DIR, Y	LDX dd, Y	41_{16}, 05_{16}, dd	3	5
ABS	LDX mmll	07_{16}, ll, mm	3	3
ABS, Y	LDX mmll, Y	41_{16}, 06_{16}, ll, mm	4	5

Note : In the immediate addressing mode, the byte number inclease by 1 when flag x = "0."

Description example:

```
CLM
LDX.W    #IMM16    ; X ← IMM16
LDX      MEM16     ; X ← MEM16
SEM
LDX.B    #IMM8     ; XL ← IMM8
LDX      MEM8      ; XL ← MEM8
```

Function : Load

Operation data length: 16 bits

Operation : X ← IMM8 (Extension zero)

X

00_{16} | | ← IMM8

Description : Extends the 8-bit immediate value to the 16-bit immediate value with 0s, and loads the data to X.

- This instruction is unaffected by flag x.
- The contents of X_H are always set to "00_{16}."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	—

N : Always "0" because MSB of the operation result is "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDXB #imm	27_{16}, imm	2	1

Description example:

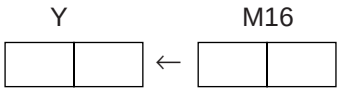
```
LDXB    #IMM8        ; X ← IMM8 (XH ← 00₁₆, XL ← IMM8)
```

Function : Load

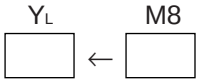
Operation data length: 16 bits or 8 bits

Operation : Y ← M

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Loads the contents of a memory to Y.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDY #imm	$D6_{16}$, imm	2	1
DIR	LDY dd	12_{16}, dd	2	3
DIR, X	LDY dd, X	41_{16}, $1B_{16}$, dd	3	5
ABS	LDY mmll	17_{16}, ll, mm	3	3
ABS, X	LDY mmll, X	41_{16}, $1F_{16}$, ll, mm	4	5

Note : In the immediate addressing mode, the byte number inclease by 1 when flag x = "0."

Description example:

```
CLM
LDY.W    #IMM16    ; Y ← IMM16
LDY      MEM16     ; Y ← MEM16
SEM
LDY.B    #IMM8     ; YL ← IMM8
LDY      MEM8      ; YL ← MEM8
```

Function : Load

Operation data length: 16 bits

Operation : Y ← IMM8 (Extension zero)

Y

00_{16}		← IMM8

Description : Extends the 8-bit immediate value to the 16-bit immediate value with 0s, and loads the data to Y.

- This instruction is unaffected by flag x.
- The contents of Y_H are always set to "00_{16}."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	—

N : Always "0" because MSB of the operation result is "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	LDYB #imm	37_{16}, imm	2	1

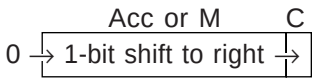
Description example:

```
LDYB        #IMM8                ; Y ← IMM8 (YH ← 00₁₆, YL ← IMM8)
```

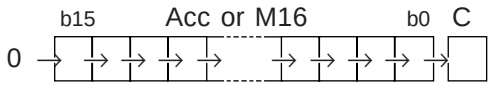
Function : Logical shift to the right

Operation data length: 16 bits or 8 bits

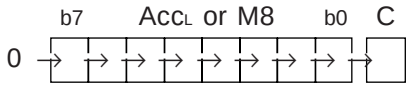
Operation :



When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Shifts all bits of Acc or a memory to the right by 1 bit. In this time, “0” is placed in MSB of Acc or a memory. Flag C is loaded from LSB of the data before the shift.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	C

N : Cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when LSB before the operation is “1.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	LSR A	43_{16}	1	1
A	LSR B	81_{16}, 43_{16}	2	2
DIR	LSR dd	21_{16}, $2A_{16}$, dd	3	7
DIR, X	LSR dd, X	21_{16}, $2B_{16}$, dd	3	8
ABS	LSR mmll	21_{16}, $2E_{16}$, ll, mm	4	7
ABS, X	LSR mmll, X	21_{16}, $2F_{16}$, ll, mm	4	8

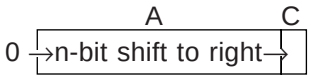
Description example:

```
CLM
LSR     A       ; A ← A is logically shifted to the right by 1 bit.
LSR     MEM16   ; MEM16 ← MEM16 is logically shifted to the right by 1 bit.
SEM
LSR     A       ; AL ← AL is logically shifted to the right by 1 bit.
LSR     MEM8    ; MEM8 ← MEM8 is logically shifted to the right by 1 bit.
```

Function : Logical shift to the right

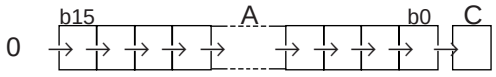
Operation data length: 16 bits or 8 bits

Operation :

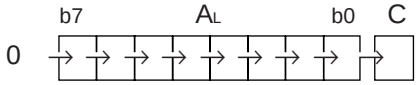


(n : Number of times shifted. n = 0 to 15)

When m = "0"



When m = "1"



❋ In this case, the contents of A_H do not change.

Description : Shifts all bits of A to the right by n bits. A "0" is placed in MSB of A, and LSB is placed in flag C each time its contents shifted by 1 bit.

● B cannot be used in this instruction.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	C

N : Always "0" because MSB of the operation result is "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" if LSB = "1" when the contents of A are shifted by (n – 1) bits. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	LSR A, #imm	$C1_{16}$, imm	2	imm+6

Note : Any value (number of times shifted) from 0 to 15 can be set to imm.

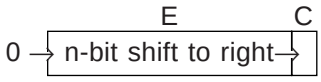
Description example:

```
CLM
LSR     A, #15      ; A ← A is logically shifted to the right by 15 bits.
SEM
LSR     A, #7       ; AL ← AL is logically shifted to the right by 7 bits.
```

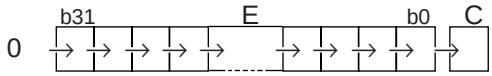
Function : Logical shift to the right

Operation data length: 32 bits

Operation :



(n : Number of times shifted. n = 0 to 31)



Description : Shifts all bits of E in 32-bit length to the right by n bits. A "0" is placed in MSB of E, and LSB is placed in flag C each time its contents are shifted by 1 bit.

- This instruction is unaffected by flag m.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	0	—	—	—	—	—	Z	C

N : Always "0" because MSB of the operation result is "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Set to "1" if LSB = "1" when the contents of E are shifted by (n – 1) bits. Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	LSRD E, #imm	$D1_{16}$, imm	2	imm+8

Note : Any value (number of times shifted) from 0 to 31 can be set to imm.

Description example:

```
LSRD    E, #16          ; E ← E is logically shifted to the right by 16 bits.
```

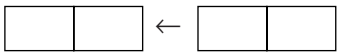
Function : Move memory to memory

Operation data length: 16 bits or 8 bits

Operation : M ← M

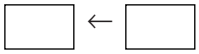
When m = "0"

M16(dest) M16(source)



When m = "1"

M8(dest) M8(source)



Description : Transfers the contents of the source memory to the destination memory.

● This instruction includes the function of the LDM instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode		Syntax	Machine code	Bytes	Cycles
dest	source				
DIR	IMM	MOVM dd, #imm	86_{16}, imm, dd	3	5
DIR	ABS	MOVM dd, mmll	$5C_{16}$, ll, mm, dd	4	6
DIR	ABS, X	MOVM dd, mmll, X	$5D_{16}$, ll, mm, dd	4	7
ABS	IMM	MOVM mmll, #imm	96_{16}, imm, ll, mm	4	4
ABS	DIR	MOVM mmll, dd	78_{16}, dd, ll, mm	4	5
ABS	DIR, X	MOVM mmll, dd, X	79_{16}, dd, ll, mm	4	6
ABS, X	IMM	MOVM mmll, X, #imm	31_{16}, 57_{16}, imm, ll, mm	5	6
ABS	ABS	MOVM $mmll_1$, $mmll_2$	$7C_{16}$, ll_2, mm_2, ll_1, mm_1	5	5
DIR, X	IMM	MOVM dd, X, #imm	31_{16}, 47_{16}, imm, dd	4	7
DIR	DIR	MOVM dd_1, dd_2	58_{16}, dd_2, dd_1	3	6

Note : In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

Description example:

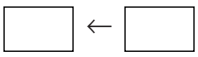
```
CLM
MOVM.W    MEM16, #IMM16                ; MEM16 ← IMM16
MOVM      MEM16(dest), MEM16(source)   ; MEM16(dest) ← MEM16(source)
SEM
MOVM.B    MEM8, #IMM8                  ; MEM8 ← IMM8
MOVM      MEM8(dest), MEM8(source)     ; MEM8(dest) ← MEM8(source)
```

Function : Move memory to memory

Operation data length: 8 bits

Operation : M8 ← M8

M8(dest) M8(source)



Description : Transfers the contents of the source memory to the destination memory in 8-bit length.

- The contents of the source memory do not change.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode		Syntax	Machine code	Bytes	Cycles
dest	source				
DIR	IMM	MOVMB dd, #imm	$A9_{16}$, imm, dd	3	5
DIR	ABS	MOVMB dd, mmll	$4C_{16}$, ll, mm, dd	4	6
DIR	ABS, X	MOVMB dd, mmll, X	$4D_{16}$, ll, mm, dd	4	7
ABS	IMM	MOVMB mmll, #imm	$B9_{16}$, imm, ll, mm	4	4
ABS	DIR	MOVMB mmll, dd	68_{16}, dd, ll, mm	4	5
ABS	DIR, X	MOVMB mmll, dd, X	69_{16}, dd, ll, mm	4	6
ABS, X	IMM	MOVMB mmll, X, #imm	31_{16}, $3B_{16}$, imm, ll, mm	5	6
ABS	ABS	MOVMB $mmll_1$, $mmll_2$	$6C_{16}$, ll_2, mm_2, ll_1, mm_1	5	5
DIR, X	IMM	MOVMB dd, X, #imm	31_{16}, $3A_{16}$, imm, dd	4	7
DIR	DIR	MOVMB dd_1, dd_2	48_{16}, dd_2, dd_1	3	6

Description example:

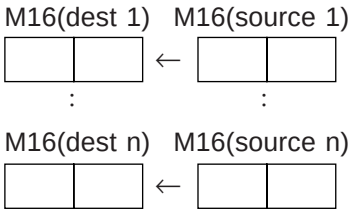
```
MOVMB   MEM8, #IMM8              ; MEM8 ← IMM8
MOVMB   MEM8(dest), MEM8(source) ; MEM8(dest) ← MEM8(source)
```

Function : Move memory to memory

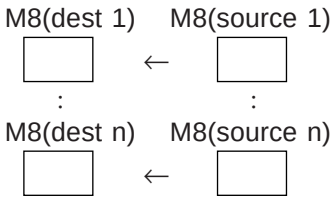
Operation data length: 16 bits or 8 bits

Operation : M(dest 1) ← M(source 1) (n : Number of times repeated transferring. n = 0 to 15)
M(dest 2) ← M(source 2)
: :
M(dest n) ← M(source n)

When m = "0"



When m = "1"



Description : Performs multiple memory-to-memory transfers by 1 instruction. Transfers are performed according to the addresses specified in the third and following bytes of the instruction. Up to 15 transfers can be performed.

- Memory contents on the source side do not change.
- No transfer is performed if a "0" is specified for the transfer count.
- This instruction can specify the different addressing modes for the source and destination, respectively; these addressing modes, however, cannot be changed until the multiple transfer specified by 1 instruction is completed.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode		Syntax	Machine code	Bytes	Cycles
dest	source				
DIR	IMM	MOVR #n, dd_1, #imm_1 , …, dd_n, #imm_n	61_{16}, n+10_{16}, imm_1, dd_1, …, imm_n, dd_n	2Xn+2 (Notes 2)	5Xn+3
DIR	DIR	MOVR #n, dd_{d1}, dd_{s1} , …, dd_{dn}, dd_{sn}	61_{16}, n+50_{16}, dd_{s1}, dd_{d1}, …, dd_{sn}, dd_{dn}	2Xn+2	6Xn+3
DIR	ABS	MOVR #n, dd_1, $mmll_1$, …, dd_n, $mmll_n$	61_{16}, n+90_{16}, ll_1, mm_1, dd_1 , …, ll_n, mm_n, dd_n	3Xn+2	6Xn+3
DIR	ABS, X	MOVR #n, dd_1, $mmll_1$, X , …, dd_n, $mmll_n$, X	71_{16}, n+10_{16}, ll_1, mm_1, dd_1 , …, ll_n, mm_n, dd_n	3Xn+2	6Xn+3
ABS	IMM	MOVR #n, $mmll_1$, #imm_1 , …, $mmll_n$, #imm_n	61_{16}, n+30_{16}, imm_1, ll_1, mm_1 , …, imm_n, ll_n, mm_n	3Xn+2 (Notes 2)	4Xn+3
ABS	DIR	MOVR #n, $mmll_1$, dd_1 , …, $mmll_n$, dd_n	61_{16}, n+70_{16}, dd_1, ll_1, mm_1 , …, dd_n, ll_n, mm_n	3Xn+2	5Xn+3
ABS	DIR, X	MOVR #n, $mmll_1$, dd_1, X , …, $mmll_n$, dd_n, X	71_{16}, n+70_{16}, dd_1, ll_1, mm_1 , …, dd_n, ll_n, mm_n	3Xn+2	6Xn+3
ABS	ABS	MOVR #n, $mmll_{d1}$, $mmll_{s1}$, …, $mmll_{dn}$, $mmll_{sn}$	61_{16}, n+$B0_{16}$, ll_{s1}, mm_{s1}, ll_{d1}, mm_{d1} , …, ll_{sn}, mm_{sn}, ll_{dn}, mm_{dn}	4Xn+2	5Xn+3

Notes 1 : Any value from 0 to 15 can be set to n.
2 : Incremented by n bytes when flag m = "0."

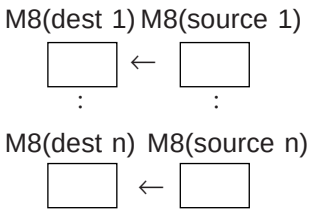
Description example:

```
CLM
MOVR.W    2, MEM16(dest1), #IMM16a, MEM16(dest2), #IMM16b
                                                    ; MEM16(dest1) ← IMM16a
                                                    ; MEM16(dest2) ← IMM16b
MOVR      2, MEM16(dest1), MEM16(source1), MEM16(dest2), MEM16(source2)
                                                    ; MEM16(dest1) ← MEM16(source1)
                                                    ; MEM16(dest2) ← MEM16(source2)
SEM
MOVR.B    2, MEM8(dest1), #IMM8a, MEM8(dest2), #IMM8b    ; MEM8(dest1) ← IMM8a
                                                    ; MEM8(dest2) ← IMM8b
MOVR      2, MEM8(dest1), MEM8(source1), MEM8(dest2), MEM8(source2)
                                                    ; MEM8(dest1) ← MEM8(source1)
                                                    ; MEM8(dest2) ← MEM8(source2)
```

Function : Move memory to memory

Operation data length: 8 bits

Operation : M8(dest 1) ← M8(source 1) (n : Number of times repeated transferring. n = 0 to 15)
M8(dest 2) ← M8(source 2)
: :
M8(dest n) ← M8(source n)



Description : Performs multiple memory-to-memory transfers by 1 instruction. Transfers are performed according to the addresses specified in the 3rd and following bytes of the instruction, in byte length. Up to 15 transfers can be performed.

- Memory contents on the source side do not change.
- No transfer is performed if a "0" is specified for the transfer count.
- This instruction can specify the different addressing modes for the source and destination, respectively; these addressing modes, however, cannot be changed until the multiple transfer specified by 1 instruction is completed.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode		Syntax	Machine code	Bytes	Cycles
dest	source				
DIR	IMM	MOVRB #n, dd_1, #imm_1 , …, dd_n, #imm_n	61_{16}, n+00_{16}, imm_1, dd_1, …, imm_n, dd_n	2×n+2	5×n+3
DIR	DIR	MOVRB #n, dd_{d1}, dd_{s1} , …, dd_{dn}, dd_{sn}	61_{16}, n+40_{16}, dd_{s1}, dd_{d1}, …, dd_{sn}, dd_{dn}	2×n+2	6×n+3
DIR	ABS	MOVRB #n, dd_1, mmll$_1$, …, dd_n, mmll$_n$	61_{16}, n+80_{16}, ll_1, mm_1, dd_1 , …, ll_n, mm_n, dd_n	3×n+2	6×n+3
DIR	ABS, X	MOVRB #n, dd_1, mmll$_1$, X , …, dd_n, mmll$_n$, X	71_{16}, n+00_{16}, ll_1, mm_1, $dd_1$3×n+2 , …, ll_n, mm_n, dd_n	6×n+3	
ABS	IMM	MOVRB #n, mmll$_1$, #imm_1 , …, mmll$_n$, #imm_n	61_{16}, n+20_{16}, imm_1, ll_1, $mm_1$3×n+2 , …, imm_n, ll_n, mm_n	4×n+3	
ABS	DIR	MOVRB #n, mmll$_1$, dd_1 , …, mmll$_n$, dd_n	61_{16}, n+60_{16}, dd_1, ll_1, mm_1 , …, dd_n, ll_n, mm_n	3×n+2	5×n+3
ABS	DIR, X	MOVRB #n, mmll$_1$, dd_1, X , …, mmll$_n$, dd_n, X	71_{16}, n+60_{16}, dd_1, ll_1, $mm_1$3×n+2 , …, dd_n, ll_n, mm_n	6×n+3	
ABS	ABS	MOVRB #n, mmll$_{d1}$, mmll$_{s1}$, …, mmll$_{dn}$, mmll$_{sn}$	61_{16}, n+$A0_{16}$, ll_{s1}, mm_{s1}, ll_{d1}, mm_{d1} , …, ll_{sn}, mm_{sn}, ll_{dn}, mm_{dn}	4×n+2	5×n+3

Note : Any value from 0 to 15 can be set to n.

Description example:

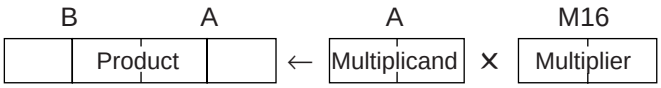
```
MOVRB   2, MEM8(dest1), #IMM8a, MEM8(dest2), #IMM8b   ; MEM8(dest1) ← IMM8a
                                                      ; MEM8(dest2) ← IMM8b
MOVRB   2, MEM8(dest1), MEM8(source1), MEM8(dest2), MEM8(source2)
                                                      ; MEM8(dest1) ← MEM8(source1)
                                                      ; MEM8(dest2) ← MEM8(source2)
```

Function : Multiplication (Unsigned)

Operation data length: 16 bits or 8 bits

Operation : (B, A) ← A (Multiplicand) × M (Multiplier)

When m = "0"



When m = "1"

❋ In this case, the contents of A_H and B_H do not change.

Description : The contents of A are multiplied by the contents of a memory. The higher of result is stored in B and lower is stored in A.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	0

N : Set to "1" when MSB (MSB of B) of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	MPY #imm	31_{16}, $C7_{16}$, imm	3	8
DIR	MPY dd	21_{16}, CA_{16}, dd	3	9
DIR, X	MPY dd, X	21_{16}, CB_{16}, dd	3	10
(DIR)	MPY (dd)	21_{16}, $C0_{16}$, dd	3	11
(DIR, X)	MPY (dd, X)	21_{16}, $C1_{16}$, dd	3	12
(DIR), Y	MPY (dd), Y	21_{16}, $C8_{16}$, dd	3	12
L(DIR)	MPY L(dd)	21_{16}, $C2_{16}$, dd	3	13
L(DIR), Y	MPY L(dd), Y	21_{16}, $C9_{16}$, dd	3	14
SR	MPY nn, S	21_{16}, $C3_{16}$, nn	3	10
(SR), Y	MPY (nn, S), Y	21_{16}, $C4_{16}$, nn	3	13
ABS	MPY mmll	21_{16}, CE_{16}, ll, mm	4	9
ABS, X	MPY mmll, X	21_{16}, CF_{16}, ll, mm	4	10
ABS, Y	MPY mmll, Y	21_{16}, $C6_{16}$, ll, mm	4	10
ABL	MPY hhmmll	21_{16}, CC_{16}, ll, mm, hh	5	10
ABL, X	MPY hhmmll, X	21_{16}, CD_{16}, ll, mm, hh	5	11

Notes 1: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

2: The cycle number in this table applies to the case of 8-bit × 8-bit operation. In the case of 16-bit × 16-bit operation, the cycle number increases by 4.

Description example:

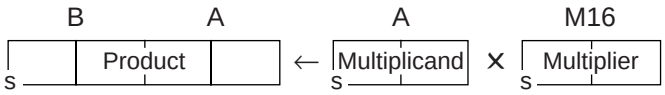
```
CLM
MPY.W    #IMM16      ; B, A ← A × IMM16
MPY      MEM16       ; B, A ← A × MEM16
SEM
MPY.B    #IMM8       ; BL, AL ← AL × IMM8
MPY      MEM8        ; BL, AL ← AL × MEM8
```

Function : Multiplication (Signed)

Operation data length: 16 bits or 8 bits

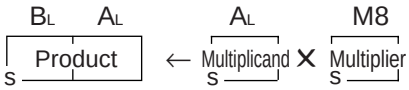
Operation : (B, A) ← A (Multiplicand) × M (Multiplier)

When m = "0"



❋ S represents MSB of the data.

When m = "1"



❋ S represents MSB of the data.

❋ In this case, the contents of A_H and B_H do not change.

Description : The contents of A are multiplied by the contents of a memory. The high order of result is stored in B and low order is stored in A. MSB of B becomes the sign bit.

Status flags

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	0

N : Set to "1" when MSB (MSB of B) of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	MPYS #imm	31_{16}, $D7_{16}$, imm	3	8
DIR	MPYS dd	21_{16}, DA_{16}, dd	3	9
DIR, X	MPYS dd, X	21_{16}, DB_{16}, dd	3	10
(DIR)	MPYS (dd)	21_{16}, $D0_{16}$, dd	3	11
(DIR, X)	MPYS (dd, X)	21_{16}, $D1_{16}$, dd	3	12
(DIR), Y	MPYS (dd), Y	21_{16}, $D8_{16}$, dd	3	12
L(DIR)	MPYS L(dd)	21_{16}, $D2_{16}$, dd	3	13
L(DIR), Y	MPYS L(dd), Y	21_{16}, $D9_{16}$, dd	3	14
SR	MPYS nn, S	21_{16}, $D3_{16}$, nn	3	10
(SR), Y	MPYS (nn, S), Y	21_{16}, $D4_{16}$, nn	3	13
ABS	MPYS mmll	21_{16}, DE_{16}, ll, mm	4	9
ABS, X	MPYS mmll, X	21_{16}, DF_{16}, ll, mm	4	10
ABS, Y	MPYS mmll, Y	21_{16}, $D6_{16}$, ll, mm	4	10
ABL	MPYS hhmmll	21_{16}, DC_{16}, ll, mm, hh	5	10
ABL, X	MPYS hhmmll, X	21_{16}, DD_{16}, ll, mm, hh	5	11

Notes 1: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

2: The cycle number in this table applies to the case of 8-bit × 8-bit operation. In the case of 16-bit × 16-bit operation, the cycle number increases by 4.

Description example:

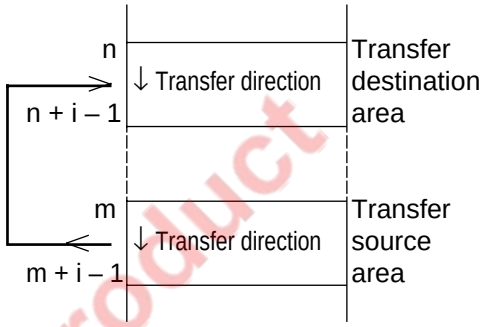
```
CLM
MPYS.W    #IMM16        ; B, A ← A × IMM16
MPYS      MEM16         ; B, A ← A × MEM16
SEM
MPYS.B    #IMM8         ; BL, AL ← AL × IMM8
MPYS      MEM8          ; BL, AL ← AL × MEM8
```

Function : Move

Operation data length: 16 bits or 8 bits

Operation : M (n to n + i – 1) ← M (m to m + i – 1) (i : transfer byte number)

Description : Normally, a block of data is transferred from higher addresses to lower addresses. The transfer is performed in the ascending address order of the block being transferred.



- The 3rd byte of the instruction : Transfer destination bank,
  The 4th byte of the instruction : Transfer source bank,
  X : Transfer destination address,
  Y : Transfer source address,
  A : Byte number of the transfed data block are specified.
  (Specify X, Y, and A before this instruction is executed.)
- When m = "0" : 0- to 65535-byte data can be transferred.
  When m = "1" : 0- to 255-byte data can be transferred.
  When x = "0" : Transfer source area and transfer destination area can be set to the addresses from 0 to 65535 ($FFFF_{16}$).
  When x = "1" : Transfer source area and transfer destination area can be set to the addresses from 0 to 255 (FF_{16}).
- Contents of registers after transfer
  X : Transfer source area end (highest) address + 1
  Y : Transfer destination area end (highest) address + 1
  A : $FFFF_{16}$
  DT : Bank number of transfer destination

Status flags :

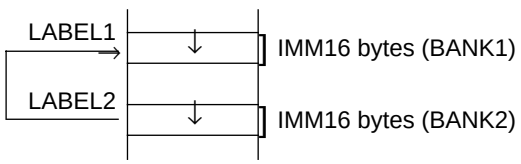
IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
BLK	MVN hh_1, hh_2	31_{16}, $2B_{16}$, hh_1, hh_2	4	5 × i + 5

Note: The cycle number in this table applies when the number of bytes transferred, i, is an even number. When i is an odd number, the cycle number is obtained as follows:
5 × i + 10.

Description example:

```
CLM                    ;
LDA.W   #IMM16         ;
LDX     LABEL2         ;
LDY     LABEL1         ;
MVN     BANK1, BANK2   ;
```

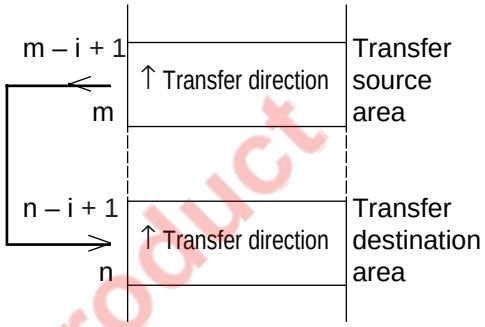


Function : Move

Operation data length: 16 bits or 8 bits

Operation : M (n – i + 1 to n) ← M (m – i + 1 to m) (i : transfer byte number)

Description : Normally, a block of data is transferred from lower addresses to higher addresses. The transfer is performed in the descending address order of the block being transferred.



- The 3rd byte of the instruction : Transfer destination bank,
  The 4th byte of the instruction : Transfer source bank,
  X : Transfer destination address,
  Y : Transfer source address,
  A : Byte number of the transfed data block are specified.
  (Specify X, Y, and A before this instruction is executed.)
- When m = "0" : 0- to 65535-byte data can be transferred.
  When m = "1" : 0- to 255-byte data can be transferred.
  When x = "0" : Transfer source area and transfer destination area can be set to the addresses from 0 to 65535 ($FFFF_{16}$).
  When x = "1" : Transfer source area and transfer destination area can be set to the addresses from 0 to 255 (FF_{16}).
- Contents of registers after transfer
  X : Transfer source area end (lowest) address – 1
  Y : Transfer destination area end (lowest) address – 1
  A : $FFFF_{16}$
  DT : Bank number of transfer destination

Status flags :

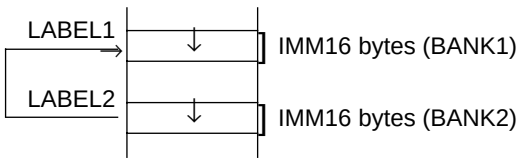
IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
BLK	MVP hh_1, hh_2	31_{16}, $2A_{16}$, hh_1, hh_2	4	5 × i + 9

Note: The cycle number in this table applies when the number of bytes transferred, i, is an even number. When i is an odd number, the cycle number is obtained as follows:
5 × i + 14 (note that the cycle number becomes 10 when 1 byte is transferred).

Description example:

```
CLM                 ;
LDA.W   #IMM16      ;
LDX     LABEL1      ;
LDY     LABEL2      ;
MVP     BANK2, BANK1;
```

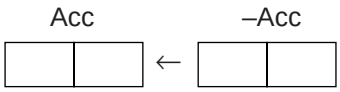


Function : Negation

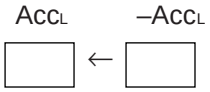
Operation data length: 16 bits or 8 bits

Operation : Acc ← –Acc

When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Negates the sign of Acc contents, and stores the result in Acc.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when the result of the operation (regarded as an unsigned operation) exceeds +65535 (+255 when flag m is “1”). Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	NEG A	24_{16}	1	1
A	NEG B	81_{16}, 24_{16}	2	2

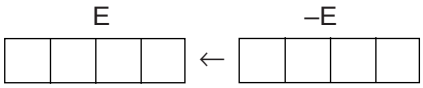
Description example:

```
CLM
NEG     A               ; A ← –A
SEM
NEG     B               ; BL ← –BL
```

Function : Negation

Operation data length: 32 bits

Operation : E ← –E



Description : Negates the sign of E contents, and stores the result in E.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when the result of the operation (regarded as an unsigned operation) exceeds +4294967295. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	NEGD E	31_{16}, 80_{16}	2	4

Description example:

```
NEGD        E                    ; E ← –E
```

Function : No operation

Operation data length: –

Operation : PC ← PC + 1
(If a carry occurs in PC, PG ← PG + 1)

Description : Only increments the program counter by 1 and nothing else.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	NOP	74_{16}	1	1

Description example:

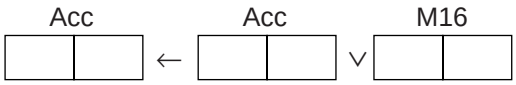
```
NOP                                    ;
```

Function : Logical OR

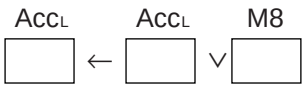
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc ∨ M

When m = "0"



When m = "1"



❋ In this case, the contents of Acc_H do not change.

Description : Performs the logical OR between the contents of Acc and the contents of a memory, and stores the result in Acc.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ORA A, #imm	56_{16}, imm (81_{16}, 56_{16}, imm)	2 (3)	1 (2)
DIR	ORA A, dd	$5A_{16}$, dd (81_{16}, $5A_{16}$, dd)	2 (3)	3 (4)
DIR, X	ORA A, dd, X	$5B_{16}$, dd (81_{16}, $5B_{16}$, dd)	2 (3)	4 (5)
(DIR)	ORA A, (dd)	11_{16}, 50_{16}, dd (91_{16}, 50_{16}, dd)	3 (3)	6 (6)
(DIR, X)	ORA A, (dd, X)	11_{16}, 51_{16}, dd (91_{16}, 51_{16}, dd)	3 (3)	7 (7)
(DIR), Y	ORA A, (dd), Y	11_{16}, 58_{16}, dd (91_{16}, 58_{16}, dd)	3 (3)	7 (7)
L(DIR)	ORA A, L(dd)	11_{16}, 52_{16}, dd (91_{16}, 52_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	ORA A, L(dd), Y	11_{16}, 59_{16}, dd (91_{16}, 59_{16}, dd)	3 (3)	9 (9)
SR	ORA A, nn, S	11_{16}, 53_{16}, nn (91_{16}, 53_{16}, nn)	3 (3)	5 (5)
(SR), Y	ORA A, (nn, S), Y	11_{16}, 54_{16}, nn (91_{16}, 54_{16}, nn)	3 (3)	8 (8)
ABS	ORA A, mmll	$5E_{16}$, ll, mm (81_{16}, $5E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	ORA A, mmll, X	$5F_{16}$, ll, mm (81_{16}, $5F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	ORA A, mmll, Y	11_{16}, 56_{16}, ll, mm (91_{16}, 56_{16}, ll, mm)	4 (4)	5 (5)
ABL	ORA A, hhmmll	11_{16}, $5C_{16}$, ll, mm, hh (91_{16}, $5C_{16}$, ll, mm, hh)	5 (5)	5 (5)
ABL, X	ORA A, hhmmll, X	11_{16}, $5D_{16}$, ll, mm, hh (91_{16}, $5D_{16}$, ll, mm, hh)	5 (5)	6 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace “A” with “B” in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = “0.”

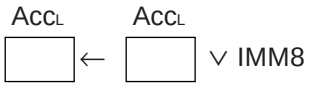
Description example:

```
CLM
ORA.W    A, #IMM16      ; A ← A ∨ IMM16
ORA      B, MEM16       ; B ← B ∨ MEM16
SEM
ORA.B    A, #IMM8       ; AL ← AL ∨ IMM8
ORA      B, MEM8        ; BL ← BL ∨ MEM8
```

Function : Logical OR

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L \vee IMM8$



Description : Performs logical OR between the contents of Acc_L and immediate value in length of 8 bits, and stores the result in Acc.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ORAB A, #imm	63_{16}, imm	2	1
IMM	ORAB B, #imm	81_{16}, 63_{16}, imm	3	2

Description example:

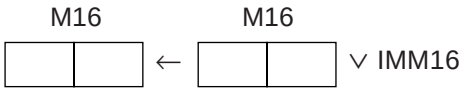
```
ORAB    A, #IMM8        ; AL ← AL ∨ IMM8
ORAB    B, #IMM8        ; BL ← BL ∨ IMM8
```

Function : Logical OR

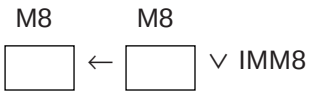
Operation data length: 16 bits or 8 bits

Operation : M← M ∨ IMM

When m = "0"



When m = "1"



Description : Performs the logical OR between the contents of a memory and the immediate value, and stores the result in the memory.

● This instruction includes the function of the SEB instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ORAM dd, #imm	51_{16}, 33_{16}, dd, imm	4	7
ABS	ORAM mmll, #imm	51_{16}, 37_{16}, ll, mm, imm	5	7

Note : When flag m = "0." the byte number increases by 1.

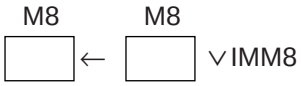
Description example:

```
CLM
ORAM.W   MEM16, #IMM16        ; MEM16 ← MEM16 ∨ IMM16
SEM
ORAM.B   MEM8, #IMM8          ; MEM8 ← MEM8 ∨ IMM8
```

Function : Logical OR

Operation data length: 8 bits

Operation : M8 ← M8 ∨ IMM8



Description : Performs the logical OR between the contents of a memory and the immediate value in 8 bits length, and stores the result in the memory.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ORAMB dd, #imm	51_{16}, 32_{16}, dd, imm	4	7
ABS	ORAMB mmll, #imm	51_{16}, 36_{16}, ll, mm, imm	5	7

Description example:

```
ORAMB    MEM8, #IMM8        ; MEM8 ← MEM8 ∨ IMM8
```

Function : Logical OR

Operation data length: 32 bits

Operation : M32 ← M32 ∨ IMM32

Description : Performs the logical OR between the contents of a memory and immediate value in 32 bits length, and stores the result in the memory.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	ORAMD dd, #imm	51_{16}, $B3_{16}$, dd, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	7	10
ABS	ORAMD mmll, #imm	51_{16}, $B7_{16}$, ll, mm, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	8	10

Description example:

```
ORAMD    MEM32, #IMM32        ; MEM32 ← MEM32 ∨ IMM32
```

Function : Stack manipulation (Push)

Operation data length: 16 bits

Operation : Stack ← IMM16

	Stack
(S) just after instruction execution	
	IMM_L
(S) just before instruction execution	IMM_H

Description : Pushes the 16-bit immediate value onto the stack.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PEA #$imm_H imm_L$	31_{16}, $4C_{16}$, imm_L, imm_H	4	5

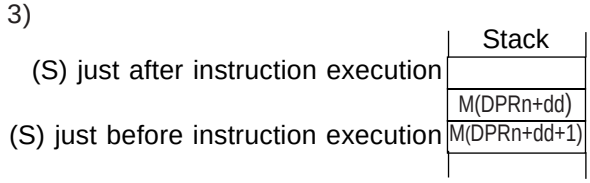
Description example:

```
PEA     #IMM16      ; (S) ← IMM16H
                    ; (S – 1) ← IMM16L
```

Function : Stack manipulation (Push)

Operation data length: 16 bits

Operation : Stack ← M16(DPRn + dd) (n = 0 to 3)



Description : Pushes the contents of the address specified by the sum of the contents of the DPRn and the offset value onto the stack in 16-bit length.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PEI dd	31_{16}, $4B_{16}$, dd	3	7

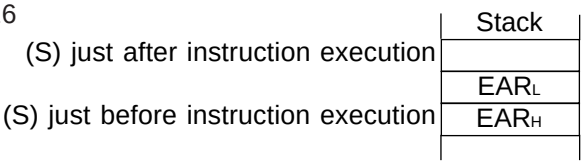
Description example:

```
PEI     DP0+: offset    ; (S) ← (DPR0 + dd + 1)
                        ; (S – 1) ← (DPR0 + dd)
```

Function : Stack manipulation (Push)

Operation data length: 16 bits

Operation : Stack ← PC + IMM16



✵ EAR = PC + IMM16

Description : Pushes the sum of the PC contents and 16-bit immediate value onto the stack in length of 16 bits.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PER #immHimmL	31_{16}, $4D_{16}$, immL, immH	4	6

Description example:

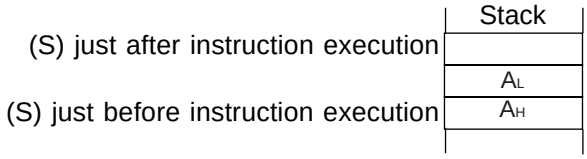
```
PER     #IMM16      ; (S) ← (PC + IMM16)H
                    ; (S – 1) ← (PC + IMM16)L
```

Function : Stack manipulation (Push)

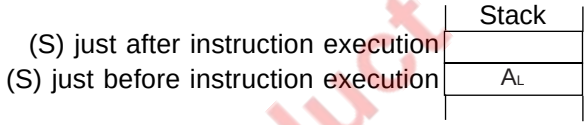
Operation data length: 16 bits or 8 bits

Operation : Stack ← A

When m = "0"



When m = "1"



Description : Pushes the contents of A onto the stack.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHA	85_{16}	1	4

Description example:

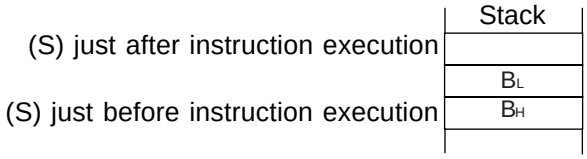
```
CLM
PHA          ; (S) ← AH, (S – 1) ← AL
SEM
PHA          ; (S) ← AL
```

Function : Stack manipulation (Push)

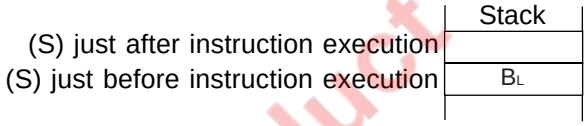
Operation data length: 16 bits or 8 bits

Operation : Stack ← B

When m = "0"



When m = "1"



Description : Pushes the contents of B onto the stack.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHB	81_{16}, 85_{16}	2	5

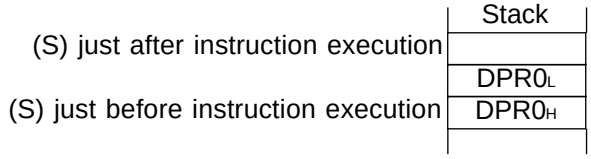
Description example:

```
CLM
PHB          ; (S) ← BH, (S – 1) ← BL
SEM
PHB          ; (S) ← BL
```

Function : Stack manipulation (Push)

Operation data length: 16 bits

Operation : Stack ← DPR0



Description : Pushes the contents of DPR0 in 16-bit length onto the stack.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHD	83_{16}	1	4

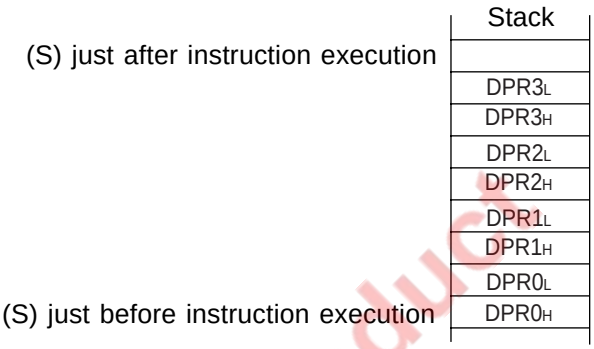
Description example:

```
PHD                    ; (S, S – 1) ← DPR0
```

Function : Stack manipulation

Operation data length: 16 bits

Operation : Stack ← DPRn (n = 0 to 3. Multiple DPRs can be pushed onto the stack.)
When DPR0 to DPR3 are specified



Description : Pushes the contents of the specified DPRn (DPR0 to DPR3) in 16-bit length onto the stack.

- Multiple DPRs can be pushed onto the stack by 1 instruction. If multiple DPRs are specified, they are pushed onto the stack in order of DPR0, DPR1, DPR2, and DPR3.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHD n	$B8_{16}$, $0?_{16}$	2	12
	PHD (n_1, …, n_i)	$B8_{16}$, $0?_{16}$	2	i + 11

Notes 1: Any value from 0 to 3 can be set to n.
2: The second line of the syntax format pushes multiple DPRs by 1 instruction.
3: The inside of parentheses (n1, …, ni) specifies 0 to 3 (numbers representing DPRn).
4: i : indicates DPRn specified (1 to 4).
5: ? : the bit corresponding to the specified DPRn becomes "1."
The diagram below shows the relationship between bits and DPRn.

b7							b0
0	0	0	0	DPR3	DPR2	DPR1	DPR0

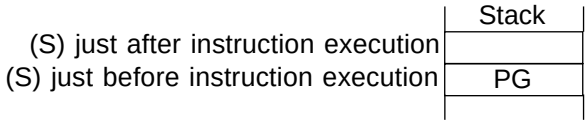
Description example:

```
PHD     1       ; (S, S – 1) ← DPR1
PHD     (0, 3)  ; (S, S – 1) ← DPR0
                ; (S – 2, S – 3) ← DPR3
```

Function : Stack manipulation (Push)

Operation data length: 8 bits

Operation : Stack ← PG



Description : Pushes the contents of PG in 8-bit length onto the stack.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHG	31_{16}, 60_{16}	2	4

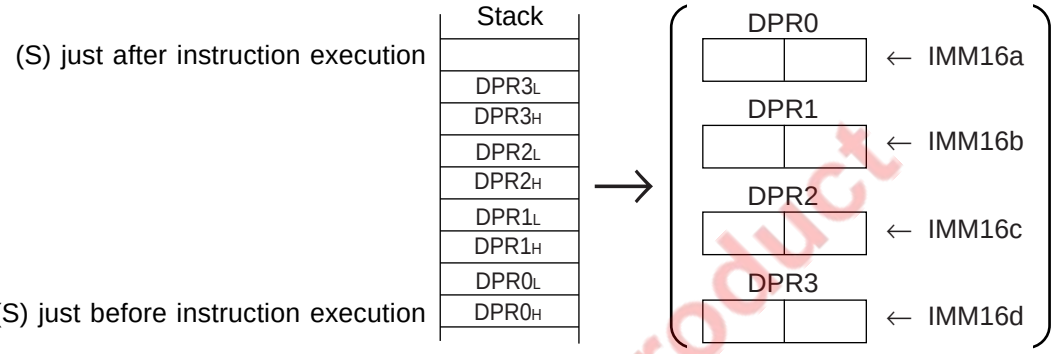
Description example:

```
PHG                    ; (S) ← PG
```

Function : Stack manipulation and Load

Operation data length: 16 bits

Operation : Stack ← DPRn (n = 0 to 3. Multiple DPRs can be specified.)
DPRn ← IMM16
When DPR0 to DPR3 are specified



Description : Loads the 16-bit immediate value to DPRn (DPR0 to DPR3), after pushing the contents of the specified DPRn in 16-bit length onto the stack.

- Multiple DPRs can be specified. If multiple DPRs are specified, they are pushed onto the stack in order of DPR0, DPR1, DPR2, and DPR3, and loads the immediate value in the same order.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHLD n, #imm	$B8_{16}$, $??_{16}$, imm_L, imm_H	4	14
	PHLD (n_1, …, n_i) , #imm_1, …, #imm_i	$B8_{16}$, $??_{16}$, imm_{L1}, imm_{H1} , …, imm_{Li}, imm_{Hi}	2 × i + 2	3 × i + 11

Notes 1: Any value from 0 to 3 can be set to n.
2: The second line of the syntax format pushes multiple DPRs by 1 instruction.
3: The inside of parentheses (n1, ···, ni) specifies 0 to 3 (numbers representing DPRn).
4: i : indicates DPRn specified (1 to 4).
5: ? : the bit corresponding to the specified DPRn becomes "1."
The diagram below shows the relationship between bits and DPRn.

b7							b0
DPR3	DPR2	DPR1	DPR0	DPR3	DPR2	DPR1	DPR0

✻ b(n) and b(n + 4) become the same contents (n = 0 to 3).

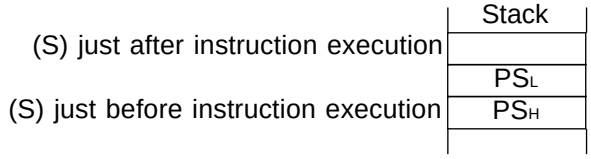
Description example:

```
PHLD    0, #IMM16                ; (S, S – 1) ← DPR0
                                 ; DPR0 ← IMM16
PHLD    (0, 3), #IMM16a, #IMM16b ; (S, S – 1) ← DPR0
                                 ; (S – 2, S – 3) ← DPR3
                                 ; DPR0 ← IMM16a
                                 ; DPR3 ← IMM16b
```

Function : Stack manipulation (Push)

Operation data length: 16 bits

Operation : Stack ← PS



Description : Pushes the contents of PS in 16-bit length onto the stack.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHP	$A5_{16}$	1	4

Description example:

```
PHP                    ; (S, S – 1) ← PS
```

Function : Stack manipulation (Push)

Operation data length: 8 bits

Operation : Stack ← DT

	Stack
(S) just after instruction execution	
(S) just before instruction execution	DT

Description : Pushes the contents of DT in 8-bit length onto the stack.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHT	31_{16}, 40_{16}	2	4

Description example:

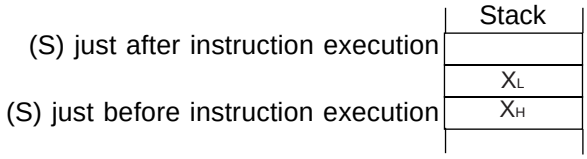
```
PHT                    ; (S) ← DT
```

Function : Stack manipulation (Push)

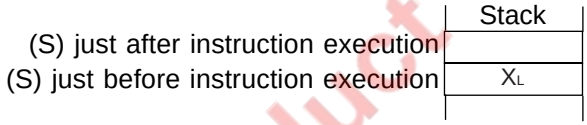
Operation data length: 16 bits or 8 bits

Operation : Stack ← X

When x = "0"



When x = "1"



Description : Pushes the contents of X onto the stack.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHX	$C5_{16}$	1	4

Description example:

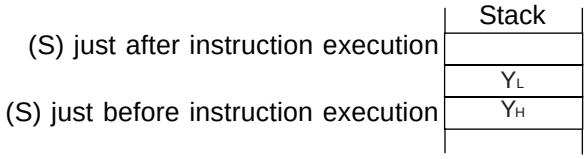
```
CLP     x
PHX                     ; (S, S – 1) ← X
SEP     x
PHX                     ; (S) ← XL
```

Function : Stack manipulation (Push)

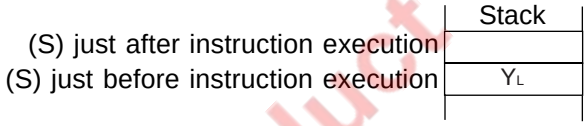
Operation data length: 16 bits or 8 bits

Operation : Stack ← Y

When x = "0"



When x = "1"



Description : Pushes the contents of Y onto the stack.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PHY	$E5_{16}$	1	4

Description example:

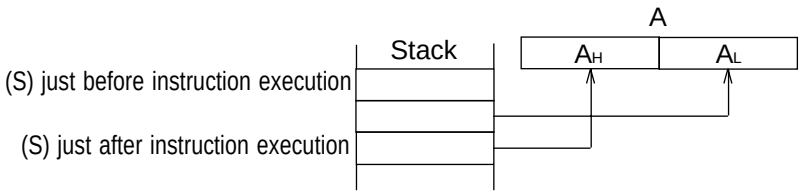
```
CLP     x
PHY                 ; (S, S – 1) ← Y
SEP     x
PHY                 ; (S) ← YL
```

Function : Stack manipulation

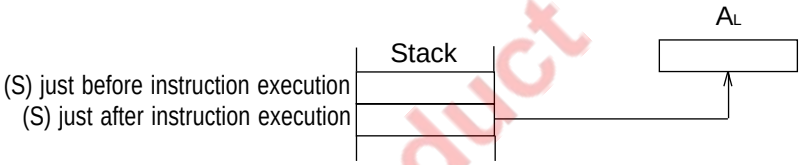
Operation data length: 16 bits or 8 bits

Operation : A ← Stack

When m = "0"



When m = "1"



❋ In this case, the contents of A_H do not change.

Description : Restores the contents of the stack to A.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLA	95_{16}	1	4

Description example:

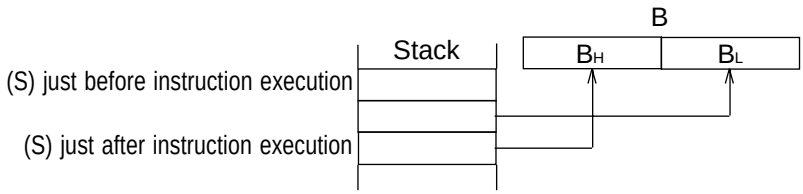
```
CLB
PLA          ; AL ← (S + 1) , AH ← (S + 2)
SEB
PLA          ; AL ← (S + 1)
```

Function : Stack manipulation

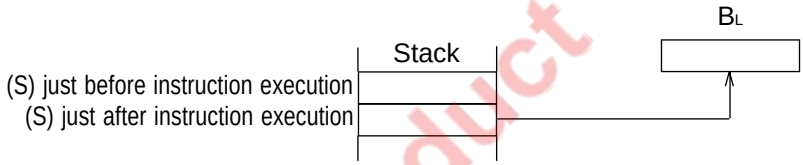
Operation data length: 16 bits or 8 bits

Operation : B ← Stack

When m = "0"



When m = "1"



❋ In this case, the contents of B_H do not change.

Description : Restores the contents of the stack to B.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLB	81_{16}, 95_{16}	2	5

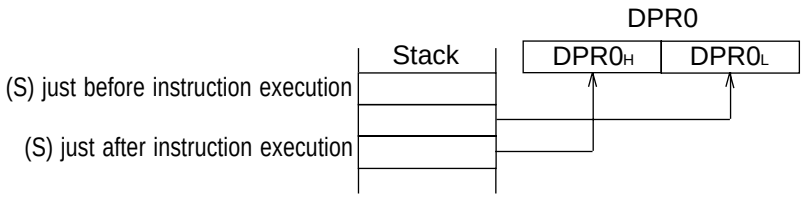
Description example:

```
CLB
PLB          ; BL ← (S + 1) , BH ← (S + 2)
SEB
PLB          ; BL ← (S + 1)
```

Function : Stack manipulation

Operation data length: 16 bits

Operation : DPR0 ← Stack



Description : Restores the contents of the stack in 16-bit length to DPR0.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLD	93_{16}	1	5

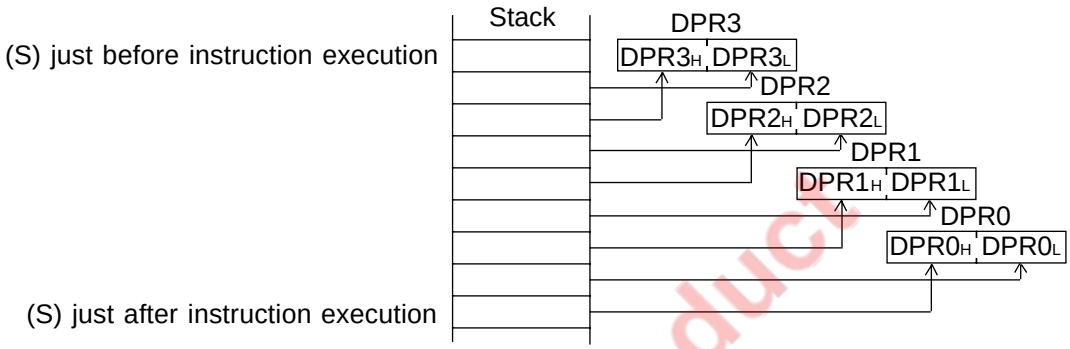
Description example:

```
PLD     ; DPR0L ← (S + 1)
        ; DPR0H ← (S + 2)
```

Function : Stack manipulation

Operation data length: 16 bits

Operation : DPRn ← Stack (n = 0 to 3. The contents of the stack can be restored to multiple DPRs.)
When DPR0 to DPR3 are specified



Description : Restores the contents of the stack to the specified DPRn (DPR0 to DPR3) in 16-bit length.

- Only 1 instruction can restore the contents of the stack to multiple DPRs. If multiple DPRs are specified, the contents of the stack are restored to DPRs in order of DPR3, DPR2, DPR1, and DPR0.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLD n	77_{16}, $?0_{16}$	2	11
	PLD (n_1, …, n_i)	77_{16}, $?0_{16}$	2	3 × i + 8

Notes 1: Any value from 0 to 3 can be set to n.
2: The second line of the syntax format restores the contents of the stack to multiple DPRs by 1 instruction.
3: Inside of the parentheses (n1, …, ni) specifies 0 to 3 (numbers representing DPRn).
4: i : indicates the number of the DPRn specified (1 to 4)
5: ? : the bit corresponding to the specified DPRn becomes "1."
The diagram below shows the relationship between bits and DPRn.

b7							b0
DPR3	DPR2	DPR1	DPR0	0	0	0	0

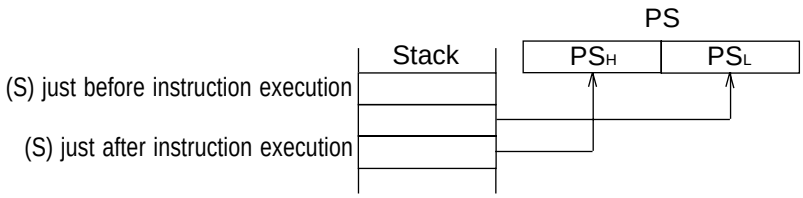
Description example:

```
PLD     1       ; DPR1 ← (S + 1, S + 2)
PLD     (0, 3)  ; DPR3 ← (S + 1, S + 2)
                ; DPR0 ← (S + 3, S + 4)
```

Function : Stack manipulation

Operation data length: 16 bits

Operation : PS ← Stack



Description : Restores the contents of the stack in 16-bit length to PS.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
IPL	N	V	m	x	D	I	Z	C

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLP	$B5_{16}$	1	5

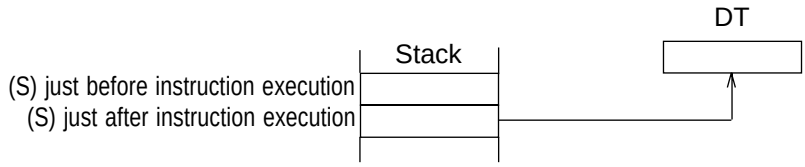
Description example:

```
PLP          ; PSL ← (S + 1)
             ; PSH ← (S + 2)
```

Function : Stack manipulation

Operation data length: 8 bits

Operation : DT ← Stack



Description : Restores the contents of the stack in 8-bit length to DT.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLT	31_{16}, 50_{16}	2	6

Description example:

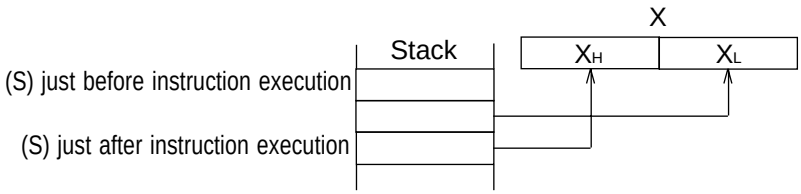
```
PLT                    ; DT ← (S + 1)
```

Function : Stack manipulation

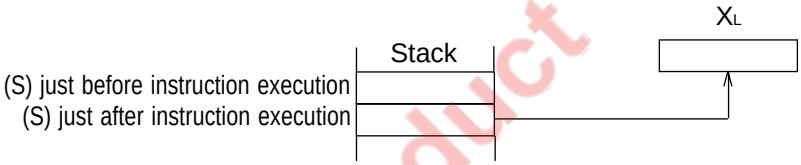
Operation data length: 16 bits or 8 bits

Operation : X ← Stack

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Restores the contents of the stack to X.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLX	$D5_{16}$	1	4

Description example:

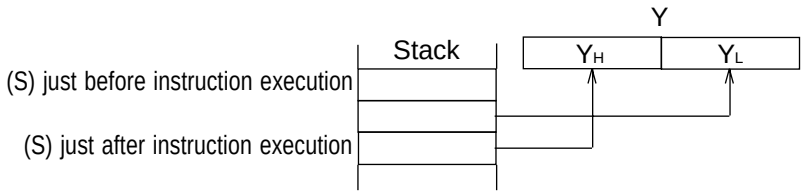
```
CLP   x
PLX                ; XL ← (S + 1) , XH ← (S + 2)
SEP   x
PLX                ; XL ← (S + 1)
```

Function : Stack manipulation

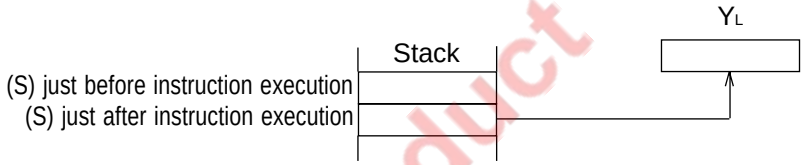
Operation data length: 16 bits or 8 bits

Operation : Y ← Stack

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Restores the contents of the stack to Y.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PLY	$F5_{16}$	1	4

Description example:

```
CLP     x
PLY                    ; YL ← (S + 1) , YH ← (S + 2)
SEP     x
PLY                    ; YL ← (S + 1)
```

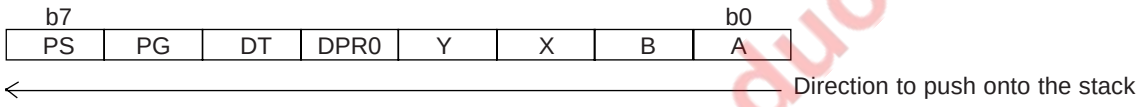
Function : Stack manipulation

Operation data length: 16 bits or 8 bits

Operation : Stack ← Specified registers among A, B, X, Y, DPR0, DT, PG, PS (Multiple registers can be specified.)
M(S to S – i + 1) ← A, B, X, Y, DPR0, DT, PG, PS
S ← S – i

i : Number of bytes corresponding to the registers pushed onto the stack.

Description : Pushes the contents of the specified registers onto the stack. Specified registers to be pushed are indicated with the bit pattarn of the 8-bit immediate value. The contents of the registers corresponding to the bits set to "1" are pushed onto the stack.



- When m = "0" : A and (or) B are (is) pushed in 16-bit length.
  When m = "1" : A_L and (or) B_L are (is) pushed in 8-bit length.
- When x = "0" : X and (or) Y are (is) pushed in 16-bit length.
  When x = "1" : X_L and (or) Y_L are (is) pushed in 8-bit length.
- This instruction is unaffected by the flags m and x when the contents of PS, PG, DT, and DPR0 are pushed onto the stack.

Status flags :

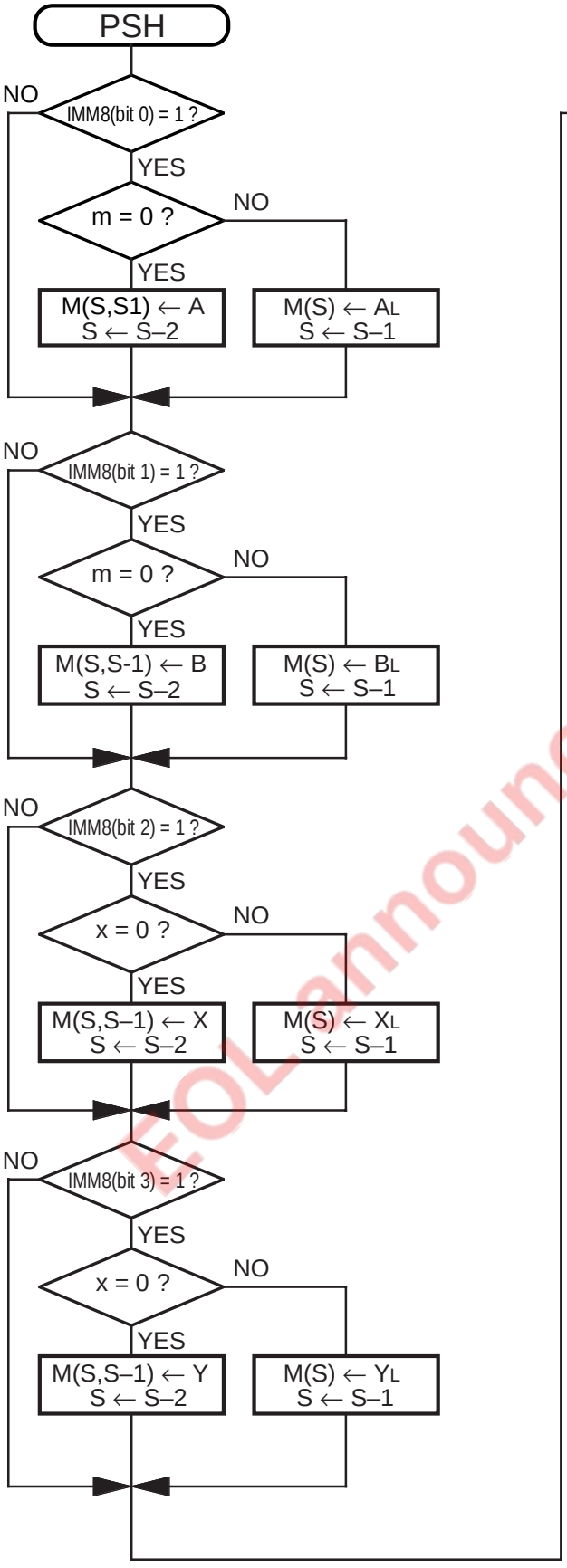
IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

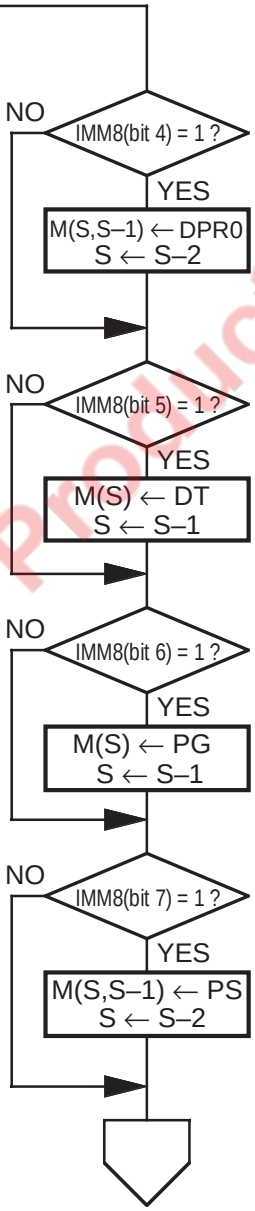
Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PSH #imm	$A8_{16}$, imm	2	$2Xi_1+i_2+11$

Notes i_1 **:** Number of registers to be pushed is indicated among A, B, X, Y, DPR0 and PS.
i_2 **:** Number of registers to be pushed DT and PG.

Description example:

```
PSH     #IMM8          ; (S) ← Contents of specified register
```





❊ IMM8 is a 1-byte immediate value, and the inside of ( ) indicates the bit position.

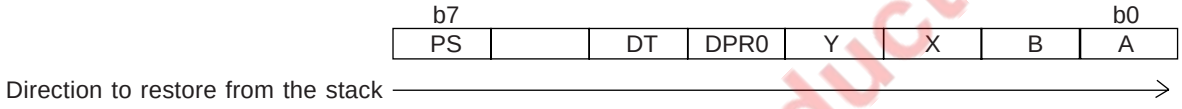
Function : Stack manipulation

Operation data length: 16 bits or 8 bits

Operation : Specified registers among A, B, X, Y, DPR0, DT, PS (Multiple registers can be specified.) ← Stack
A, B, X, Y, DPR0, DT, PS ← M(S + 1 to S + i)
S ← S + i

i : Number of bytes corresponding to the registers restored from the stack.

Description : Restores the stack contents to the specified registers. Specified registers to be restored are indicated with the bit pattarn of the 8-bit immediate value. The stack contents are restored to the registers corresponding to the bits that are set to "1."



- When m of restored PS = "0" : Restored to A and (or) B in 16-bit length.
  When m of restored PS = "1" : Restored to A_L and (or) B_L in 8-bit length.
  In this case, the contents of A_H and B_H do not change.
- When x of restored PS = "0" : Restored to X and (or) Y in 16-bit length.
  When x of restored PS = "1" : Restored to X_L and (or) Y_L in 8-bit length.
  In this case, the contents of X_H and Y_H do not change.

Status flags :

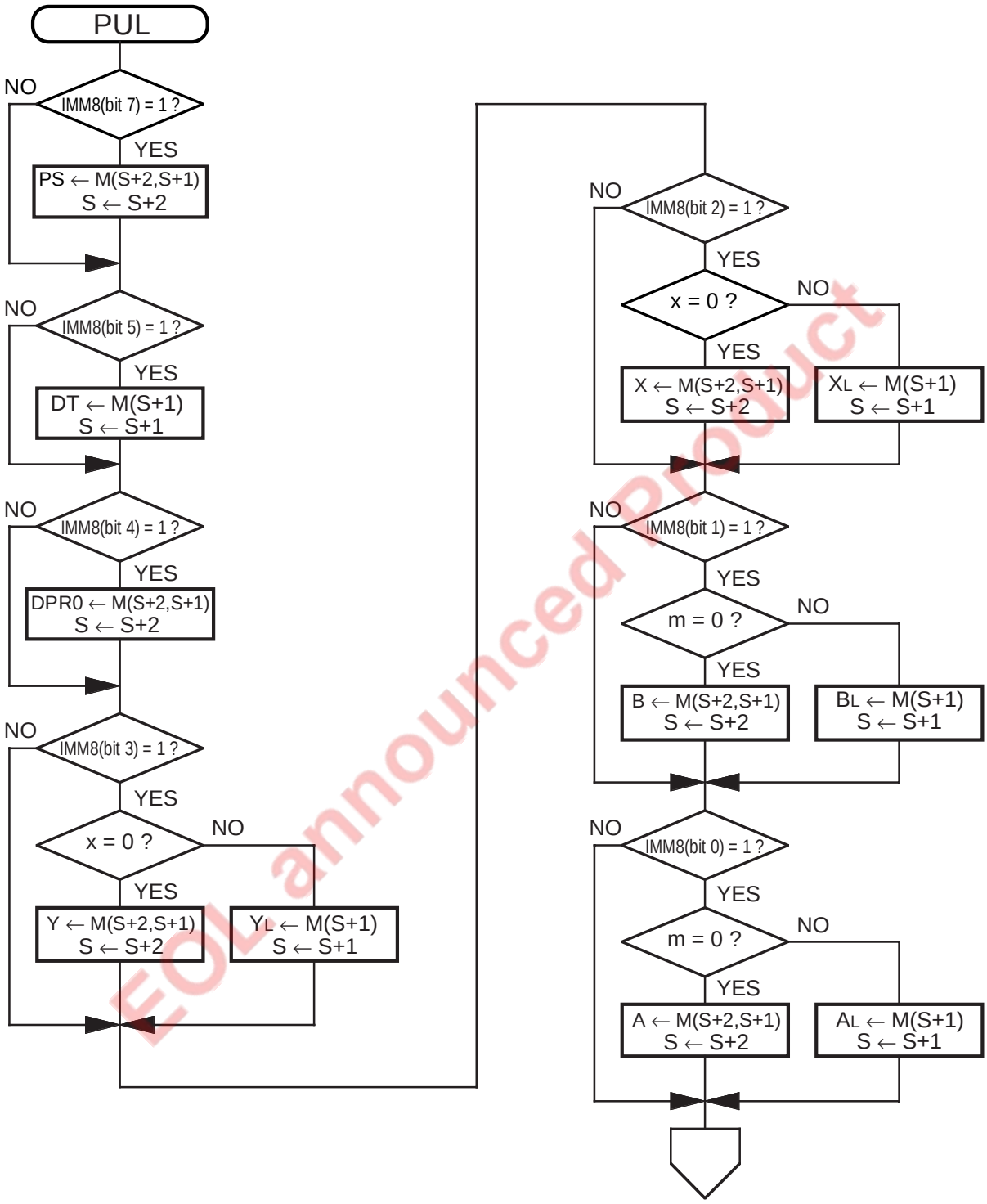
IPL	N	V	m	x	D	I	Z	C
IPL	N	V	m	x	D	I	Z	C

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	PUL #imm	67_{16}, imm	2	3Xi+13

Note i : Number of registers to be restored.

Description example:

```
PUL     #IMM8          ; Contents of specified register ← (S + 1)
```

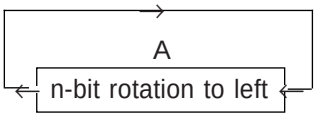


❊ IMM8 is a 1-byte immediate value, and the inside of ( ) indicates the bit position.

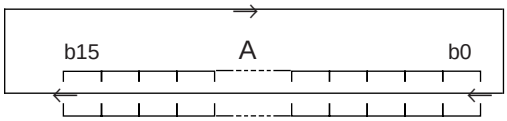
Function : Rotation to the left

Operation data length: 16 bits or 8 bits

Operation :



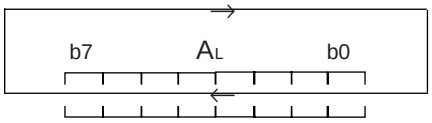
When m = “0”



n-bit rotation to the left

❋ n = 0 to 65535

When m = “1”



n-bit rotation to the left

❋ n = 0 to 255

❋ In this case, the contents of A_H do not change.

Description : Rotates the contents of A to the left by n bits.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	RLA #imm	31_{16}, 07_{16}, imm	3	n + 5

Notes 1: n : Indicates the number of rotation specified by imm.
2: When flag m = “0,” the byte number increases by 1.

Description example:

```
CLM
RLA     #IMM16      ; A ← A is rotated to the left according to the times specified by IMM16.
SEM
RLA     #IMM8       ; AL ← AL is rotated to the left according to the times specified by IMM8.
```

Function : Multiplied accumulation repeated

Operation data length: 16 bits or 8 bits

Operation : (B, A) ← (B, A) + M (DT:X) × M (DT:Y) (repeated 0 to 255 times.)

Description : Performs signed multiplication between the contents of addresses specified by the contents of X and Y in the bank indicated by DT. Then, the multiplication result is added to the contents of B and A respectively, and these addition results are stored in B and A; and the contents of X and Y each are incremented. This operation is repeated as many times (0 to 255 times) as specified by the 8-bit immediate value in the third byte of this instruction.

- When m = "0" : Operates in 16-bit length, and the result becomes the 32-bit value.
  E ← E + M16 (DT:X) × M16 (DT:Y)
  After the addition, the contents of X and Y each are incremented by 2.
- When m = "1" : Operates in 8-bit length, and the result becomes the 16-bit value.
  (B_L, A_L) ← (B_L, A_L) + M8 (DT:X) × M8 (DT:Y)
  In this case, the contents of A_H and B_H do not change.
  After the addition, the contents of X and Y each are incremented by 1.
- Contents of X and Y after operation: The addresses next to those of the multiplicand and multiplier which were read out last, respectively.
- If an overflow occurs as an addition result, the flag V is set to "1" and the operation finishes halfway. In this time, the contents of A and B become undefined. The contents of X and Y become the addresses next to those of the multiplicand and multiplier which were read out last, respectively.
- The instruction is terminated without performing any operation if a "0" is specified for the repeat count. In this case, the contents of A, B, X, and Y do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : This flag is checked for each addition performed. If MSB (MSB of B) of the addition result becomes "1," this flag becomes "1." Otherwise, cleared to "0."

V : This flag is checked for each addition performed. If the addition result is a value outside the range of –2147483648 to +2147483647 (or –32768 to +32767 when flag m = "1"), this flag is set to "1." Otherwise, cleared to "0." If flag V = "0" when the instruction is terminated, it means that the operation has terminated normally; if flag V = "1," it means that an overflow occured.

Z : This flag is checked for each addition performed. Set to "1," when the addition result becomes "0." Otherwise, cleared to "0."

C : This flag is checked for each addition performed. Set to "1" when the addition result (regarded as an unsigned data) exceeds +4294967295 (or +65536 when flag m = "1"). Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
Multiplied accumulation	RMPA #imm	31_{16}, $5A_{16}$, imm	3	14×imm+5

Notes 1: imm ; indicates the number of repeated operation.
2: The cycle number in this table applies when flag m = "1." When flag m = "0," the cycle number becomes 18 × imm + 5.

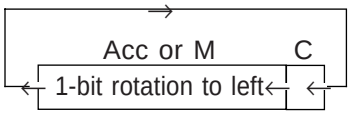
Description example:

```
RMPA    #IMM8          ; repeates the operation IMM8 times.
```

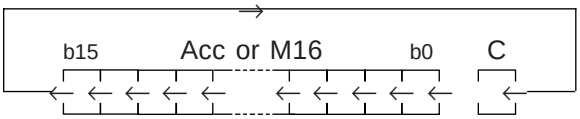
Function : Rotation to the left

Operation data length: 16 bits or 8 bits

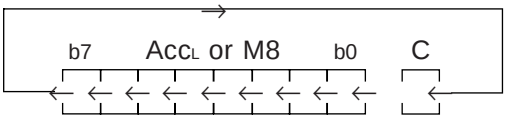
Operation :



When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Flag C is linked to Acc or a memory, and the combined contents are rotated to the left by 1 bit.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when MSB of the data before rotation is “1.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ROL A	13_{16}	1	1
A	ROL B	81_{16}, 13_{16}	2	2
DIR	ROL A, dd	21_{16}, $1A_{16}$, dd	3	7
DIR, X	ROL A, dd, X	21_{16}, $1B_{16}$, dd	3	8
ABS	ROL A, mmll	21_{16}, $1E_{16}$, ll, mm	4	7
ABS, X	ROL A, mmll, X	21_{16}, $1F_{16}$, ll, mm	4	8

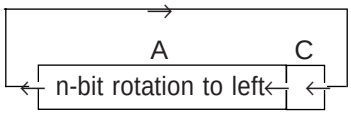
Description example:

```
CLM
ROL     A        ; A is rotated to the left by 1 bit.
ROL     MEM16    ; MEM16 is rotated to the left by 1 bit.
SEM
ROL     B        ; BL is rotated to the left by 1 bit.
ROL     MEM16    ; MEM8 is rotated to the left by 1 bit.
```

Function : Rotation to the left

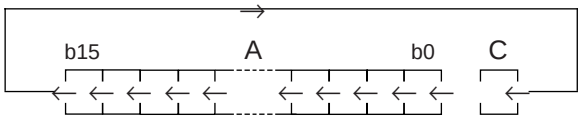
Operation data length: 16 bits or 8 bits

Operation :

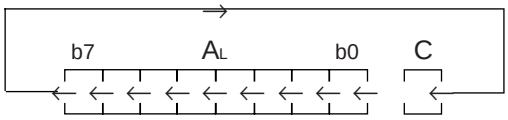


(n : times of rotation. n = 0 to 15)

When m = “0”



When m = “1”



❋ In this case, the contents of A_H do not change.

Description : Flag C is linked to A, and the combined contents are rotated to the left by n bits.

● B cannot be used in this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” if MSB = “1” when the contents are rotated by (n – 1) bits. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ROL A, #imm	$C1_{16}$, imm+60_{16}	2	imm + 6

Note: Any value from 0 to 15 (times of rotation) can be set to imm.

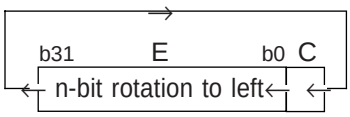
Description example:

```
CLM
ROL     A, #15      ; A ← A combined with C is rotated to the left by 15 bits.
SEM
ROL     A, #7       ; AL ← AL combined with C is rotated to the left by 7 bits.
```

Function : Rotation to the left

Operation data length: 32 bits

Operation :



(n : times of rotation. n = 0 to 31)

Description : Flag C is linked to E, and the combined contents are rotated to the left by n bits in 32-bit length.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” if MSB = “1” when the contents are rotated by (n–1) bits. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ROLD E, #imm	$D1_{16}$, imm+60_{16}	2	imm + 8

Note: Any value from 0 to 31 (times of rotation) can be set to imm.

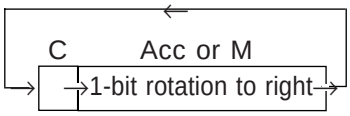
Description example:

```
ROLD    E, #16          ; E ← E combined with C is rotated to the left by 16 bits.
```

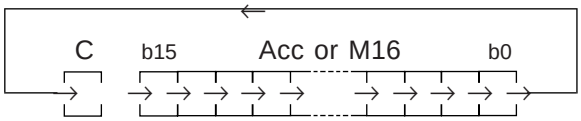
Function : Rotation to the right

Operation data length: 16 bits or 8 bits

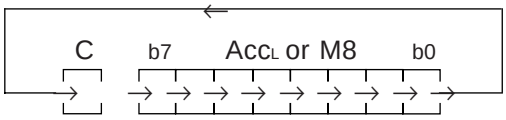
Operation :



When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Flag C is linked to Acc or a memory, and the combined contents are rotated to the right by 1 bit.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” when LSB of the data before rotation is “1.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ROR A	53_{16}	1	1
A	ROR B	81_{16}, 53_{16}	2	2
DIR	ROR A, dd	21_{16}, $3A_{16}$, dd	3	7
DIR, X	ROR A, dd, X	21_{16}, $3B_{16}$, dd	3	8
ABS	ROR A, mmll	21_{16}, $3E_{16}$, ll, mm	4	7
ABS, X	ROR A, mmll, X	21_{16}, $3F_{16}$, ll, mm	4	8

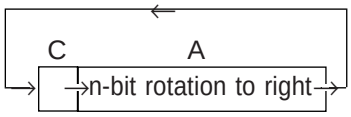
Description example:

```
CLM
ROR     A          ; A is rotated to the right by 1 bit.
ROR     MEM16      ; MEM16 is rotated to the right by 1 bit.
SEM
ROR     B          ; BL is rotated to the right by 1 bit.
ROR     MEM8       ; MEM8 is rotated to the right by 1 bit.
```

Function : Rotation to the right

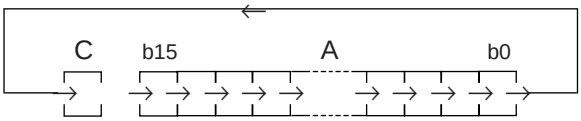
Operation data length: 16 bits or 8 bits

Operation :

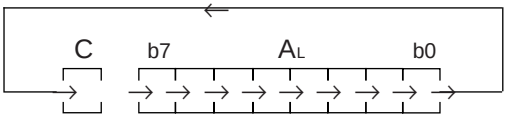


(n : times of rotation. n = 0 to 15)

When m = “0”



When m = “1”



❋ In this case, the contents of A_H do not change.

Description : Flag C is linked to A, and the combined contents are rotated to the right by n bits.

● B cannot be used in this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” if LSB = “1” when the contents are rotated by (n – 1) bits. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	ROR A, #imm	$C1_{16}$, imm+20_{16}	2	imm + 6

Note: Any value from 0 to 15 (times of rotation) can be set to imm.

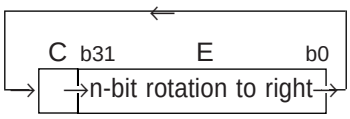
Description example:

```
CLM
ROR     A, #15          ; A ← A combined with C is rotated to the right by 15 bits.
SEM
ROR     A, #7           ; AL ← AL combined with C is rotated to the right by 7 bits.
```

Function : Rotation to the right

Operation data length: 32 bits

Operation :



(n : times of rotation. n = 0 to 31)

Description : Flag C is linked to E, and the combined contents are rotated to the right by n bits in 32-bits length.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Set to “1” if LSB = “1” when the contents are rotated by (n–1) bits. Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
A	RORD E, #imm	$D1_{16}$, imm+20_{16}	2	imm + 8

Note: Any value from 0 to 31 (times of rotation) can be set to imm.

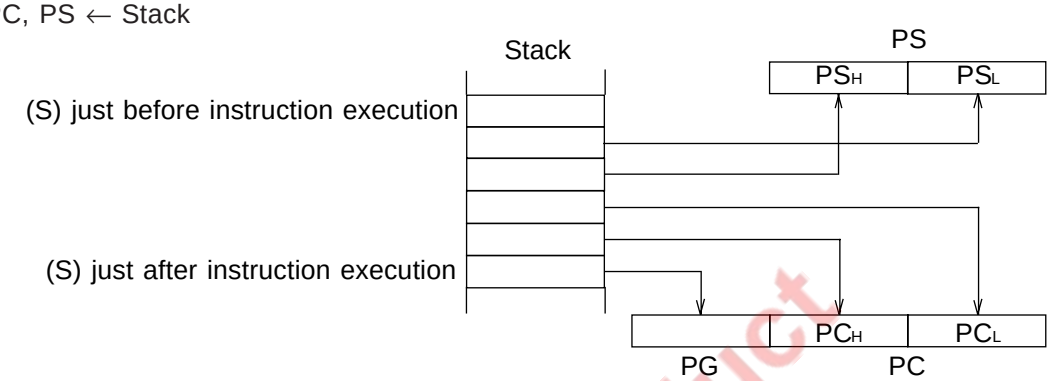
Description example:

```
RORD    E, #16          ; E ← E combined with C is rotated to the right by 16 bits.
```

Function : Return

Operation data length: –

Operation : PG, PC, PS ← Stack



Description : Restores the stack contents to the registers in order of PS, PC, and PG.

- Use this instruction when returning from the interrupt routine.

Status flags :

IPL	N	V	m	x	D	I	Z	C
IPL	N	V	m	x	D	I	Z	C

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	RTI	$F1_{16}$	1	12

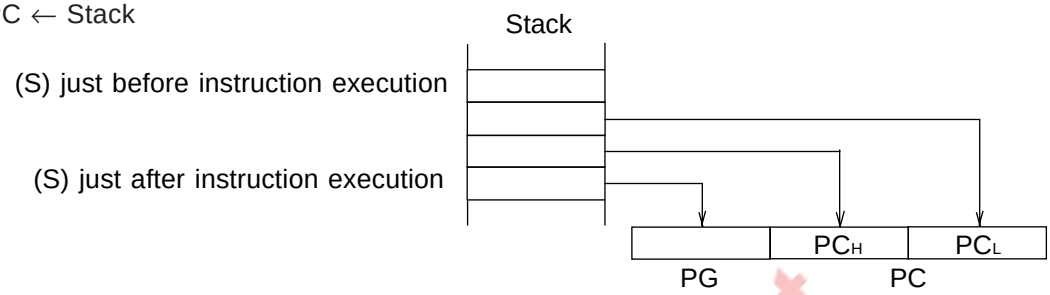
Description example:

```
RTI          ; PS ← (S + 2, S + 1)
             ; PC ← (S + 4, S + 3)
             ; PG ← (S + 5)
```

Function : Return

Operation data length: –

Operation : PG, PC ← Stack



Description : Restores the stack contents to the registers in order of PC and PG.

- Use this instruction when returning from the subroutine called by JSRL.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	RTL	94_{16}	1	10

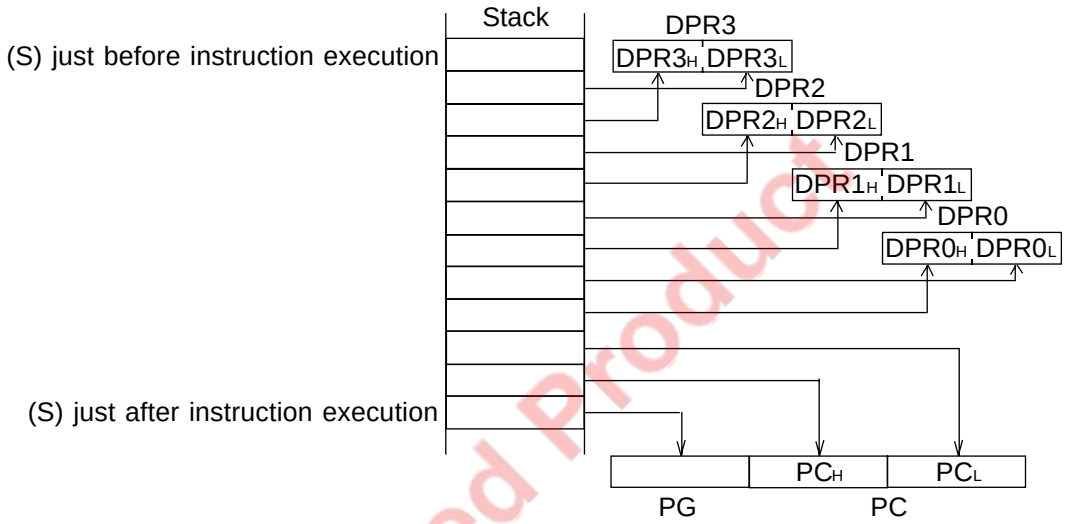
Description example:

```
RTL                    ; PC ← (S + 2, S + 1)
                       ; PG ← (S + 3)
```

Function : Load & Return

Operation data length: 16 bits

Operation : DPRn ← Stack (n = 0 to 3. Multiple DPRs can be specified.)
PG, PC ← Stack
When DPR0 to DPR3 are specified



Description : After restoring the contents of the specified DPRn (DPR0 to DPR3) from the stack in length of 16 bits, this instruction executes the RTL instruction (to restore the stack contents in order of PC and PG).

- Multiple DPRs can be specified for restoration from the stack. When multiple DPRs are specified, the stack contents are restored to DPRs in order of DPR3, DPR2, DPR1, and DPR0.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	RTLD n	77_{16}, ?C_{16}	2	15
	RTLD (n_1, …, n_i)	77_{16}, ?C_{16}	2	3 X i + 12

Notes 1: Any value from 0 can be set to 3 to n.
2: The second line of the syntax format specifies multiple DPRs by 1 instruction.
3: Inside of the parentheses (n1, …, ni) specifies any of 0 to 3 (numbers representing DPRn).
4: i : indicates the number of DPRn (1 to 4)
5: ? : the bit corresponding to the specified DPRn becomes “1.”
The diagram below shows the relationship between bits and DPRn.

b7							b0
DPR3	DPR2	DPR1	DPR0	1	1	0	0

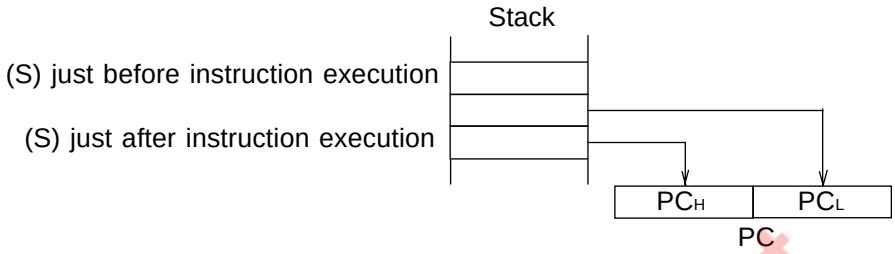
Description example:

```
RTLD    1       ; DPR1 ← (S + 1)
                ; RTL
RTLD    (0, 3)  ; DPR3 ← (S + 1)
                ; DPR0 ← (S + 3)
                ; RTL
```

Function : Return

Operation data length: –

Operation : PC ← Stack



Description : Restores the stack contents to PC.

- Use this instruction when returning from the subroutine called by JSR or BSR.
- If this instruction is located at a bank's highest address ($XXFFFF_{16}$), the contents of PG are incremented by 1.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	RTS	84_{16}	1	7

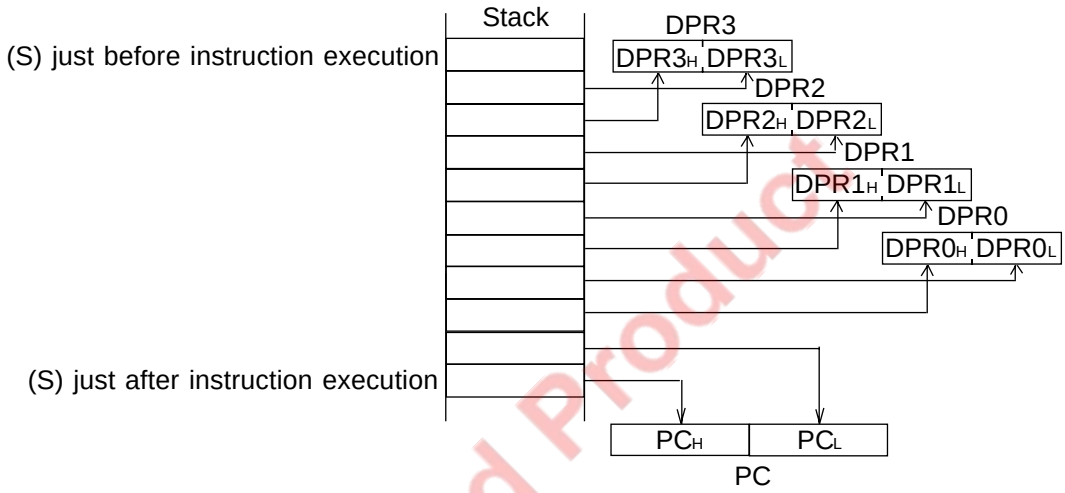
Description example:

```
RTS                                    ; PC ← (S + 2, S + 1)
```

Function : Load & Return

Operation data length: 16 bits

Operation : DPRn ← Stack (n = 0 to 3. Multiple registers can be specified.)
PC ← Stack
When DPR0 to DPR3 are specified



Description : After restoring the contents of the specified DPRn (DPR0 to DPR3) from the stack in length of 16 bits, this instruction executes the RTS instruction (to restore the stack contents to PC).

- Multiple DPRs can be specified for return from the stack. When multiple DPRs are specified, the stack contents are restored to DPRs respectively, in order of DPR3, DPR2, DPR1, and DPR0.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
STK	RTSD n	77_{16}, ?8_{16}	2	14
	RTSD (n_1, …, n_i)	77_{16}, ?8_{16}	2	3 X i +11

Notes 1: Any value from 0 to 3 can be set to n.
2: The second line of the syntax format specifies multiple DPRs by 1 instruction.
3: Inside of the parentheses (n1, …, ni) specifies any of 0 to 3 (numbers representing DPRn).
4: i : indicates the number of DPRn (1 to 4)
5: ? : the bit corresponding to the specified DPRn becomes “1.”
The diagram below shows the relationship between bits and DPRn.

b7							b0
DPR3	DPR2	DPR1	DPR0	1	0	0	0

Description example:

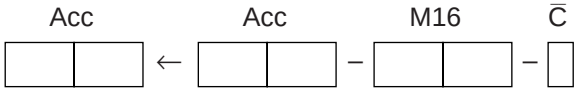
```
RTSD    1           ; DPR1 ← (S + 1)
                    ; RTS
RTSD    (0, 3)      ; DPR3 ← (S + 1)
                    ; DPR0 ← (S + 3)
                    ; RTS
```

Function : Subtract with carry

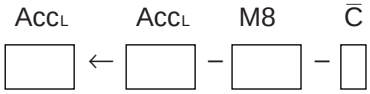
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc – M – $\overline{C}$

When m = "0"



When m = "1"



❋ In this case, the contents of Acc_H do not change.

Description : Subtracts the contents of a memory and the complement of flag C from the contents of Acc, and stores the result in Acc.

● The decimal operation is performed when flag D = "1."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is "1"). Otherwise, cleared to "0." Meaningless when flag D = "1."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0." Meaningless when flag D = "1."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SBC A, #imm	31_{16}, $A7_{16}$, imm ($B1_{16}$, $A7_{16}$, imm)	3	3 (3)
DIR	SBC A, dd	21_{16}, AA_{16}, dd ($A1_{16}$, AA_{16}, dd)	3	5 (7)
DIR, X	SBC A, dd, X	21_{16}, AB_{16}, dd ($A1_{16}$, AB_{16}, dd)	3	6 (8)
(DIR)	SBC A, (dd)	21_{16}, $A0_{16}$, dd ($A1_{16}$, $A0_{16}$, dd)	3	7 (9)
(DIR, X)	SBC A, (dd, X)	21_{16}, $A1_{16}$, dd ($A1_{16}$, $A1_{16}$, dd)	3	8 (10)
(DIR), Y	SBC A, (dd), Y	21_{16}, $A8_{16}$, dd ($A1_{16}$, $A8_{16}$, dd)	3	8 (10)
L(DIR)	SBC A, L(dd)	21_{16}, $A2_{16}$, dd ($A1_{16}$, $A2_{16}$, dd)	3	9 (11)
L(DIR), Y	SBC A, L(dd), Y	21_{16}, $A9_{16}$, dd ($A1_{16}$, $A9_{16}$, dd)	3	10(12)
SR	SBC A, nn, S	21_{16}, $A3_{16}$, nn ($A1_{16}$, $A3_{16}$, nn)	3	6 (8)
(SR), Y	SBC A, (nn, S), Y	21_{16}, $A4_{16}$, nn ($A1_{16}$, $A4_{16}$, nn)	3	9 (11)
ABS	SBC A, mmll	21_{16}, AE_{16}, ll, mm ($A1_{16}$, AE_{16}, ll, mm)	4	5 (7)
ABS, X	SBC A, mmll, X	21_{16}, AF_{16}, ll, mm ($A1_{16}$, AF_{16}, ll, mm)	4	6 (8)
ABS, Y	SBC A, mmll, Y	21_{16}, $A6_{16}$, ll, mm ($A1_{16}$, $A6_{16}$, ll, mm)	4	6 (8)
ABL	SBC A, hhmmll	21_{16}, AC_{16}, ll, mm, hh ($A1_{16}$, AC_{16}, ll, mm, hh)	5	6 (8)
ABL, X	SBC A, hhmmll, X	21_{16}, AD_{16}, ll, mm, hh ($A1_{16}$, AD_{16}, ll, mm, hh)	5	7 (9)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace "A" with "B" in the syntax. In this case, the machine code and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

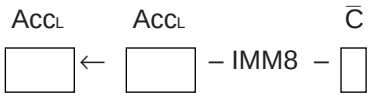
Description example:

```
CLM
SBC.W   A, #IMM16       ; A ← A – IMM16 – C̄
SBC     B, MEM16        ; B ← B – MEM16 – C̄
SEB
SBC.B   A, #IMM8        ; AL ← AL – IMM8 – C̄
SBC     B, MEM8         ; BL ← BL – MEM8 – C̄
```

Function : Subtract with carry

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L - IMM8 - \overline{C}$



Description : Subtracts the immediate value and the complement of flag C from the contents of Acc_L in 8-bit length, and stores the result in Acc_L.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.
- The decimal operation is performed when flag D = "1."

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0." Meaningless when flag D = "1."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to "0." Meaningless when flag D = "1."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0." Meaningless when flag D = "1."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	ABCB A, #imm	31_{16}, $1B_{16}$, imm	3	3
IMM	ABCB B, #imm	$B1_{16}$, $1B_{16}$, imm	3	3

Description example:

```
SBCB    A, #IMM8    ; AL ← AL – IMM8 – C̄
SBCB    B, #IMM8    ; BL ← BL – IMM8 – C̄
```

Function : Subtract with carry

Operation data length: 32 bits

Operation : $E \leftarrow E - M32 - \overline{C}$

Description : Subtracts the contents of a memory and the complement of flag C from the contents of E in 32-bit length, and stores the result in E.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SBCD E, #imm	31_{16}, $1D_{16}$, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	6	4
DIR	SBCD E, dd	21_{16}, BA_{16}, dd	3	7
DIR, X	SBCD E, dd, X	21_{16}, BB_{16}, dd	3	8
(DIR)	SBCD E, (dd)	21_{16}, $B0_{16}$, dd	3	9
(DIR, X)	SBCD E, (dd, X)	21_{16}, $B1_{16}$, dd	3	10
(DIR), Y	SBCD E, (dd), Y	21_{16}, $B8_{16}$, dd	3	10
L(DIR)	SBCD E, L(dd)	21_{16}, $B2_{16}$, dd	3	11
L(DIR), Y	SBCD E, L(dd), Y	21_{16}, $B9_{16}$, dd	3	12
SR	SBCD E, nn, S	21_{16}, $B3_{16}$, nn	3	8
(SR), Y	SBCD E, (nn, S), Y	21_{16}, $B4_{16}$, nn	3	11
ABS	SBCD E, mmll	21_{16}, BE_{16}, ll, mm	4	7
ABS, X	SBCD E, mmll, X	21_{16}, BF_{16}, ll, mm	4	8
ABS, Y	SBCD E, mmll, Y	21_{16}, $B6_{16}$, ll, mm	4	8
ABL	SBCD E, hhmmll	21_{16}, BC_{16}, ll, mm, hh	5	8
ABL, X	SBCD E, hhmmll, X	21_{16}, BD_{16}, ll, mm, hh	5	9

Description example:

```
SBCD    E, #IMM32    ; E ← E – IMM32 – C̄ (B, A ← B, A – IMM32 – C̄)
SBCD    E, MEM32     ; E ← E – MEM32 – C̄ (B, A ← B, A – MEM32 – C̄)
```

Function : Flag manipulation

Operation data length: –

Operation : C ← 1

Description : Sets flag C to “1.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	1

C : Set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	SEC	04_{16}	1	1

Description example:

```
SEC                                    ; C ← 1
```

Function : Flag manipulation

Operation data length: –

Operation : I ← 1

Description : Sets flag I to “1.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	1	—	—

I : Set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	SEI	05_{16}	1	4

Description example:

```
SEI                                    ; I ← 1
```

Function : Flag manipulation

Operation data length: –

Operation : m ← 1

Description : Sets flag m to “1.”

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	1	—	—	—	—	—

m : Set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	SEM	25_{16}	1	3

Description example:

```
SEM                                    ; m ← 1
```

Function : Flag manipulation

Operation data length: –

Operation : PS_L (bit n) ← 1 (n = 0 to 7. Multiple bits can be specified.)

Description : Sets the specified flags (multiple flags can be specified) of PS_L to "1." The flag positions to be specified are indicated by a bit pattern of the immediate value, in which the bits set to "1" are the subject bits to be specified.

- This instruction is unaffected by flag m.

PS_L

b7	b6	b5	b4	b3	b2	b1	b0
N	V	m	x	D	I	Z	C

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	m	x	D	I	Z	C

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SEP #imm	99_{16}, imm	2	3

Description example:

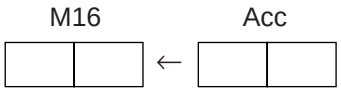
```
SEP     #IMM8          ; The specified bits of PSL ← 1
```

Function : Store

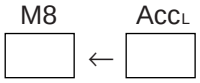
Operation data length: 16 bits or 8 bits

Operation : M ← Acc

When m = "0"



When m = "1"



Description : Stores the contents of Acc into a memory. The contents of Acc do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	STA A, dd	DA_{16}, dd (81_{16}, DA_{16}, dd)	2 (3)	4 (5)
DIR, X	STA A, dd, X	DB_{16}, dd (81_{16}, DB_{16}, dd)	2 (3)	5 (6)
(DIR)	STA A, (dd)	11_{16}, $D0_{16}$, dd (91_{16}, $D0_{16}$, dd)	3 (3)	7 (7)
(DIR, X)	STA A, (dd, X)	11_{16}, $D1_{16}$, dd (91_{16}, $D1_{16}$, dd)	3 (3)	8 (8)
(DIR), Y	STA A, (dd), Y	$D8_{16}$, dd (81_{16}, $D8_{16}$, dd)	2 (3)	7 (8)
L(DIR)	STA A, L(dd)	11_{16}, $D2_{16}$, dd (91_{16}, $D2_{16}$, dd)	3 (3)	9 (9)
L(DIR), Y	STA A, L(dd), Y	$D9_{16}$, dd (81_{16}, $D9_{16}$, dd)	2 (3)	9 (10)
SR	STA A, nn, S	11_{16}, $D3_{16}$, nn (91_{16}, $D3_{16}$, nn)	3 (3)	6 (6)
(SR), Y	STA A, (nn, S), Y	11_{16}, $D4_{16}$, nn (91_{16}, $D4_{16}$, nn)	3 (3)	9 (9)
ABS	STA A, mmll	DE_{16}, ll, mm (81_{16}, DE_{16}, ll, mm)	3 (4)	4 (5)
ABS, X	STA A, mmll, X	DF_{16}, ll, mm (81_{16}, DF_{16}, ll, mm)	3 (4)	5 (6)
ABS, Y	STA A, mmll, Y	11_{16}, $D6_{16}$, ll, mm (91_{16}, $D6_{16}$, ll, mm)	4 (4)	6 (6)
ABL	STA A, hhmmll	DC_{16}, ll, mm, hh (81_{16}, DC_{16}, ll, mm, hh)	4 (5)	5 (6)
ABL, X	STA A, hhmmll, X	DD_{16}, ll, mm, hh (81_{16}, DD_{16}, ll, mm, hh)	4 (5)	6 (7)

Note : This table applies when using accumulator A. When using accumulator B, replace "A" with "B" in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

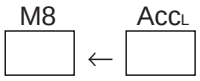
Description example:

```
CLM
STA     A, MEM16        ; MEM16 ← A
SEM
STA     B, MEM8         ; MEM8 ← BL
```

Function : Store

Operation data length: 8 bits

Operation : $M8 \leftarrow Acc_L$



Description : Stores the contents of Acc_L into a memory in 8-bit length.

- The contents of Acc (Acc_H and Acc_L) do not change.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	STAB A, dd	CA_{16}, dd (81_{16}, CA_{16}, dd)	2 (3)	4 (5)
DIR, X	STAB A, dd, X	CB_{16}, dd (81_{16}, CB_{16}, dd)	2 (3)	5 (6)
(DIR)	STAB A, (dd)	11_{16}, $C0_{16}$, dd (91_{16}, $C0_{16}$, dd)	3 (3)	7 (7)
(DIR, X)	STAB A, (dd, X)	11_{16}, $C1_{16}$, dd (91_{16}, $C1_{16}$, dd)	3 (3)	8 (8)
(DIR), Y	STAB A, (dd), Y	$C8_{16}$, dd (81_{16}, $C8_{16}$, dd)	2 (3)	7 (8)
L(DIR)	STAB A, L(dd)	11_{16}, $C2_{16}$, dd (91_{16}, $C2_{16}$, dd)	3 (3)	9 (9)
L(DIR), Y	STAB A, L(dd), Y	$C9_{16}$, dd (81_{16}, $C9_{16}$, dd)	2 (3)	9 (10)
SR	STAB A, nn, S	11_{16}, $C3_{16}$, nn (91_{16}, $C3_{16}$, nn)	3 (3)	6 (6)
(SR), Y	STAB A, (nn, S), Y	11_{16}, $C4_{16}$, nn (91_{16}, $C4_{16}$, nn)	3 (3)	9 (9)
ABS	STAB A, mmll	CE_{16}, ll, mm (81_{16}, CE_{16}, ll, mm)	3 (4)	4 (5)
ABS, X	STAB A, mmll, X	CF_{16}, ll, mm (81_{16}, CF_{16}, ll, mm)	3 (4)	5 (6)
ABS, Y	STAB A, mmll, Y	11_{16}, $C6_{16}$, ll, mm (91_{16}, $C6_{16}$, ll, mm)	4 (4)	6 (6)
ABL	STAB A, hhmmll	CC_{16}, ll, mm, hh (81_{16}, CC_{16}, ll, mm, hh)	4 (5)	5 (6)
ABL, X	STAB A, hhmmll, X	CD_{16}, ll, mm, hh (81_{16}, CD_{16}, ll, mm, hh)	4 (5)	6 (7)

Note : This table applies when using accumulator A. When using accumulator B, replace “A” with “B” in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

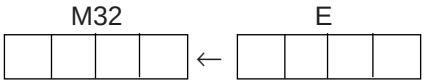
Description example:

```
STAB        A, MEM8          ; MEM8 ← AL
```

Function : Store

Operation data length: 32 bits

Operation : M32 ← E



Description : Stores the contents of E into a memory in 32-bit length.

- The contents of E do not change.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	STAD E, dd	EA_{16}, dd	2	6
DIR, X	STAD E, dd, X	EB_{16}, dd	2	7
(DIR)	STAD E, (dd)	11_{16}, $E0_{16}$, dd	3	9
(DIR, X)	STAD E, (dd, X)	11_{16}, $E1_{16}$, dd	3	10
(DIR), Y	STAD E, (dd), Y	$E8_{16}$, dd	2	9
L(DIR)	STAD E, L(dd)	11_{16}, $E2_{16}$, dd	3	11
L(DIR), Y	STAD E, L(dd), Y	$E9_{16}$, dd	2	11
SR	STAD E, nn, S	11_{16}, $E3_{16}$, nn	3	8
(SR), Y	STAD E, (nn, S), Y	11_{16}, $E4_{16}$, nn	3	11
ABS	STAD E, mmll	EE_{16}, ll, mm	3	6
ABS, X	STAD E, mmll, X	EF_{16}, ll, mm	3	7
ABS, Y	STAD E, mmll, Y	11_{16}, $E6_{16}$, ll, mm	4	8
ABL	STAD E, hhmmll	EC_{16}, ll, mm, hh	4	7
ABL, X	STAD E, hhmmll, X	ED_{16}, ll, mm, hh	4	8

Description example:

```
STAD    E, MEM32        ; MEM32 ← E (MEM32H ← B, MEM32L ← A)
```

Function : Special

Operation data length: –

Operation : Stop the oscillation

Description : Resets the flip-flop for oscillator control and stops the oscillation of the oscillation circuit. To restart, generate an interrupt request or perform the hardware reset. The microcomputer will thereby be released from the STP state.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	STP	31_{16}, 30_{16}	2	–

Description example:

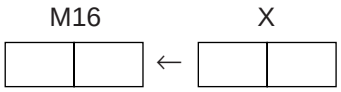
```
STP          ;
```

Function : Store

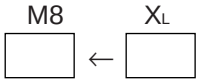
Operation data length: 16 bits or 8 bits

Operation : M ← X

When x = "0"



When x = "1"



Description : Stores the contents of X into a memory. The contents of X do not change.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	STX dd	$E2_{16}$, dd	2	4
DIR, Y	STX dd, Y	41_{16}, $E5_{16}$, dd	3	6
ABS	STX mmll	$E7_{16}$, ll, mm	3	4

Description example:

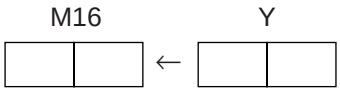
```
CLP     x
STX     MEM16          ; MEM16 ← X
SEP     x
STX     MEM8           ; MEM8 ← XL
```

Function : Store

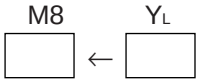
Operation data length: 16 bits or 8 bits

Operation : M ← Y

When x = "0"



When x = "1"



Description : Stores the contents of Y into a memory. The contents of Y do not change.

- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	STY dd	$F2_{16}$, dd	2	4
DIR, X	STY dd, X	41_{16}, FB_{16}, dd	3	6
ABS	STY mmll	$F7_{16}$, ll, mm	3	4

Description example:

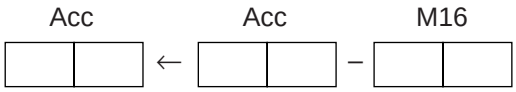
```
CLP     x
STY     MEM16          ; MEM16 ← Y
SEP     x
STY     MEM8           ; MEM8 ← YL
```

Function : Subtract

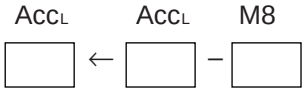
Operation data length: 16 bits or 8 bits

Operation : Acc ← Acc – M

When m = “0”



When m = “1”



❋ In this case, the contents of Acc_H do not change.

Description : Subtracts the contents of a memory from the contents of Acc, and stores the result in Acc.

● This instruction cannot operate in decimal. Set flag D = “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SUB A, #imm	36_{16}, imm (81_{16}, 36_{16}, imm)	2 (3)	1 (2)
DIR	SUB A, dd	$3A_{16}$, dd (81_{16}, $3A_{16}$, dd)	2 (3)	3 (4)
DIR, X	SUB A, dd, X	$3B_{16}$, dd (81_{16}, $3B_{16}$, dd)	2 (3)	4 (5)
(DIR)	SUB A, (dd)	11_{16}, 30_{16}, dd (91_{16}, 30_{16}, dd)	3 (3)	6 (6)
(DIR, X)	SUB A, (dd, X)	11_{16}, 31_{16}, dd (91_{16}, 31_{16}, dd)	3 (3)	7 (7)
(DIR), Y	SUB A, (dd), Y	11_{16}, 38_{16}, dd (91_{16}, 38_{16}, dd)	3 (3)	7 (7)
L(DIR)	SUB A, L(dd)	11_{16}, 32_{16}, dd (91_{16}, 32_{16}, dd)	3 (3)	8 (8)
L(DIR), Y	SUB A, L(dd), Y	11_{16}, 39_{16}, dd (91_{16}, 39_{16}, dd)	3 (3)	9 (9)
SR	SUB A, nn, S	11_{16}, 33_{16}, nn (91_{16}, 33_{16}, nn)	3 (3)	5 (5)
(SR), Y	SUB A, (nn, S), Y	11_{16}, 34_{16}, nn (91_{16}, 34_{16}, nn)	3 (3)	8 (8)
ABS	SUB A, mmll	$3E_{16}$, ll, mm (81_{16}, $3E_{16}$, ll, mm)	3 (4)	3 (4)
ABS, X	SUB A, mmll, X	$3F_{16}$, ll, mm (81_{16}, $3F_{16}$, ll, mm)	3 (4)	4 (5)
ABS, Y	SUB A, mmll, Y	11_{16}, 36_{16}, ll, mm (91_{16}, 36_{16}, ll, mm)	4 (4)	5 (5)
ABL	SUB A, hhmmll	11_{16}, $3C_{16}$, ll, mm, hh (91_{16}, $3C_{16}$, ll, mm, hh)	5 (5)	5 (5)
ABL, X	SUB A, hhmmll, X	11_{16}, $3D_{16}$, ll, mm, hh (91_{16}, $3D_{16}$, ll, mm, hh)	5 (5)	6 (6)

Notes 1: This table applies when using accumulator A. When using accumulator B, replace "A" with "B" in the syntax. In this case, the machine code, the number of bytes, and the number of cycles enclosed in parentheses are applied.

2: In the immediate addressing mode, the byte number increases by 1 when flag m = "0."

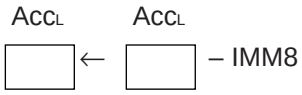
Description example:

```
CLM
SUB.W   A, #IMM16       ; A ← A – IMM16
SUB     B, MEM16        ; B ← B – MEM16
SEM
SUB.B   A, #IMM8        ; AL ← AL – IMM8
SUB     B, MEM8         ; BL ← BL – MEM8
```

Function : Subtract

Operation data length: 8 bits

Operation : $Acc_L \leftarrow Acc_L - IMM8$



Description : Subtracts the immediate value from the contents of Acc_L in 8-bit length, and stores the result in Acc_L.

- This instruction is unaffected by flag m.
- The contents of Acc_H do not change.
- This instruction cannot operate in decimal. Set flag D = "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SUBB A, #imm	39_{16}, imm	2	1
IMM	SUBB B, #imm	81_{16}, 39_{16}, imm	3	2

Description example:

```
SUBB    A, #IMM8        ; AL ← AL – IMM8
SUBB    B, #IMM8        ; BL ← BL – IMM8
```

Function : Subtract

Operation data length: 32 bits

Operation : E ← E – M32

Description : Subtracts the contents of a memory from the contents of E in 32-bit length, and stores the result in E.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SUBD E, #imm	$3D_{16}$, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	5	3
DIR	SUBD E, dd	AA_{16}, dd	2	6
DIR, X	SUBD E, dd, X	AB_{16}, dd	2	7
(DIR)	SUBD E, (dd)	11_{16}, $A0_{16}$, dd	3	9
(DIR, X)	SUBD E, (dd, X)	11_{16}, $A1_{16}$, dd	3	10
(DIR), Y	SUBD E, (dd), Y	11_{16}, $A8_{16}$, dd	3	10
L(DIR)	SUBD E, L(dd)	11_{16}, $A2_{16}$, dd	3	11
L(DIR), Y	SUBD E, L(dd), Y	11_{16}, $A9_{16}$, dd	3	12
SR	SUBD E, nn, S	11_{16}, $A3_{16}$, nn	3	8
(SR), Y	SUBD E, (nn, S), Y	11_{16}, $A4_{16}$, nn	3	11
ABS	SUBD E, mmll	AE_{16}, ll, mm	3	6
ABS, X	SUBD E, mmll, X	AF_{16}, ll, mm	3	7
ABS, Y	SUBD E, mmll, Y	11_{16}, $A6_{16}$, ll, mm	4	8
ABL	SUBD E, hhmmll	11_{16}, AC_{16}, ll, mm, hh	5	8
ABL, X	SUBD E, hhmmll, X	11_{16}, AD_{16}, ll, mm, hh	5	9

Description example:

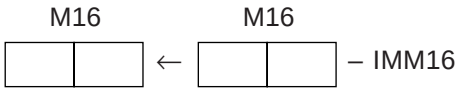
```
SUBD    E, #IMM32       ; E ← E – IMM32 (B, A ← B, A – IMM32)
SUBD    E, MEM32        ; E ← E – MEM32 (B, A ← B, A – MEM32)
```

Function : Subtract

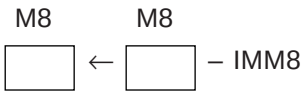
Operation data length: 16 bits or 8 bits

Operation : M← M – IMM

When m = “0”



When m = “1”



Description : Subtracts the immediate value from the contents of a memory, and stores the result in the memory.

- This instruction cannot operate in decimal. Set flag D = “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag m is “1”). Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	SUBM dd, #imm	51_{16}, 13_{16}, dd, imm	4	7
ABS	SUBM mmll, #imm	51_{16}, 17_{16}, ll, mm, imm	5	7

Note : When flag m = “0,” the byte number increases by 1.

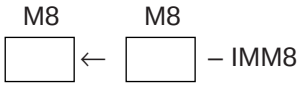
Description example:

```
CLM
SUBM.W    MEM16, #IMM16        ; MEM16 ← MEM16 – IMM16
SEM
SUBM.B    MEM8, #IMM8          ; MEM8 ← MEM8 – IMM8
```

Function : Subtract

Operation data length: 8 bits

Operation : M8 ← M8 – IMM8



Description : Subtracts the immediate value from the contents of a memory in 8-bit length, and stores the result in the memory.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –128 to +127. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	SUBMB dd, #imm	51_{16}, 12_{16}, dd, imm	4	7
ABS	SUBMB mmll, #imm	51_{16}, 16_{16}, ll, mm, imm	5	7

Description example:

```
SUBMB    MEM8, #IMM8          ; MEM8 ← MEM8 – IMM8
```

Function : Subtract

Operation data length: 32 bits

Operation : M32 ← M32 – IMM32

Description : Subtracts the immediate value from the contents of a memory in 32-bit length, and stores the result in the memory.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = “0” when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

V : Set to “1” when the result of the operation (regarded as a signed operation) is a value outside the range of –2147483648 to +2147483647. Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

C : Cleared to “0” when the borrow occurs. Otherwise, set to “1.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
DIR	SUBMD dd, #imm	51_{16}, 93_{16}, dd, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	7	10
ABS	SUBMD mmll, #imm	51_{16}, 97_{16}, ll, mm, imm_{LL}, imm_{LH}, imm_{HL}, imm_{HH}	8	10

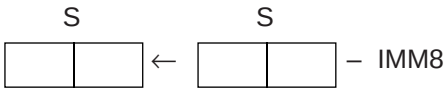
Description example:

```
SUBMB    MEM32, #IMM32          ; MEM32 ← MEM32 – IMM32
```

Function : Subtract

Operation data length: 16 bits

Operation : S ← S – IMM8



Description : Subtract the 8-bit immediate value from the contents of S in 16-bit length, and stores the result in S. The immediate value is extended to 16-bit length with 0s in operation.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767. Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SUBS #imm	31_{16}, $0B_{16}$, imm	3	2

Description example:

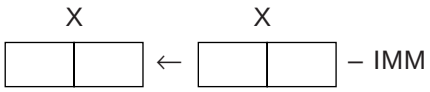
```
SUBS    #IMM8          ; S ← S – IMM8
```

Function : Subtract

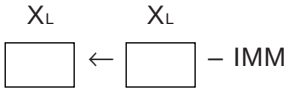
Operation data length: 16 bits or 8 bits

Operation : X ← X – IMM (IMM = 0 to 31)

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Subtracts the immediate value (0 to 31) from the contents of X, and stores the result in X.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag x is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SUBX #imm	01_{16}, imm+40_{16}	2	2

Note : Any value from 0 to 31 can be set to imm.

Description example:

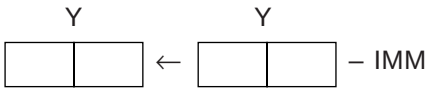
```
CLP     x
SUBX    #IMM        ; X ← X – IMM(0 to 31)
SEP     x
SUBX    #IMM        ; XL ← XL – IMM(0 to 31)
```

Function : Subtract

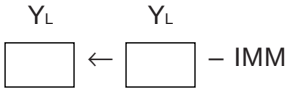
Operation data length: 16 bits or 8 bits

Operation : $Y \leftarrow Y - IMM$ (IMM = 0 to 31)

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Subtracts the immediate value (0 to 31) from the contents of Y, and stores the result in Y.

- This instruction is unaffected by flag m.
- This instruction cannot operate in decimal. Set flag D = "0" when using this instruction.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	V	—	—	—	—	Z	C

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

V : Set to "1" when the result of the operation (regarded as a signed operation) is a value outside the range of –32768 to +32767 (–128 to +127 when flag x is "1"). Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

C : Cleared to "0" when the borrow occurs. Otherwise, set to "1."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMM	SUBY #imm	01_{16}, imm+60_{16}	2	2

Note : Any value from 0 to 31 can be set to imm.

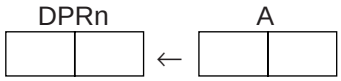
Description example:

```
CLP     x
SUBY    #IMM        ; Y ← Y – IMM(0 to 31)
SEP     x
SUBY    #IMM        ; YL ← YL – IMM(0 to 31)
```

Function : Transfer between registers

Operation data length: 16 bits

Operation : DPRn ← A (n = 0 to 3)



Description : Transfers the contents of A to the specified DPRn (DPR0 to DPR3) in 16-bit length.

- Specify one of DPR0 to DPR3 for the destination of transfer.
- The contents of A do not change.
- This instruction is unaffected by flag m.
- This instruction includes the function of the TAD instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TAD n	31_{16}, $n2_{16}$	2	3

Note : Any value from 0 to 3 can be set to n.

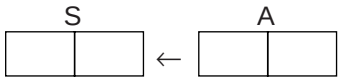
Description example:

```
TAD     0       ; DPR0 ← A
TAD     1       ; DPR1 ← A
```

Function : Transfer between registers

Operation data length: 16 bits

Operation : S ← A



Description : Transfers the contents of A to S in 16-bit length. The contents of A do not change.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TAS	31_{16}, 82_{16}	2	2

Description example:

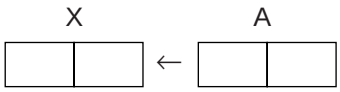
```
TAS                    ; S ← A
```

Function : Transfer between registers

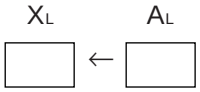
Operation data length: 16 bits or 8 bits

Operation : $X \leftarrow A$

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Transfers the contents of A to X. The contents of A do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TAX	$C4_{16}$	1	1

Description example:

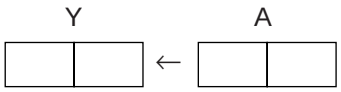
```
CLP   x
TAX                ; X ← A
SEP   x
TAX                ; XL ← AL
```

Function : Transfer between registers

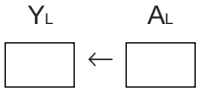
Operation data length: 16 bits or 8 bits

Operation : Y ← A

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Transfers the contents of A to Y. The contents of A do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TAY	$D4_{16}$	1	1

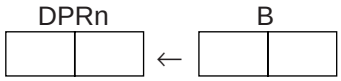
Description example:

```
CLP   x
TAY                ; Y ← A
SEP   x
TAY                ; YL ← AL
```

Function : Transfer between registers

Operation data length: 16 bits

Operation : DPRn ← B (n = 0 to 3)



Description : Transfers the contents of B to the specified DPRn (DPR0 to DPR3) in 16-bit length.

- Specify one of DPR0 to DPR3 for the destination of transfer.
- The contents of B do not change.
- This instruction is unaffected by flag m.
- This instruction includes the function of the TBD instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TBD n	$B1_{16}$, $n2_{16}$	2	3

Note : Any value from 0 to 3 can be set to n.

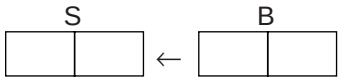
Description example:

```
TBD     0       ; DPR0 ← B
TBD     1       ; DPR1 ← B
```

Function : Transfer between registers

Operation data length: 16 bits

Operation : S ← B



Description : Transfers the contents of B to S in 16-bit length. The contents of B do not change.
- This instruction is unaffected by flag m.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TBS	$B1_{16}$, 82_{16}	2	2

Description example:

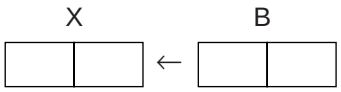
```
TBS                    ; S ← B
```

Function : Transfer between registers

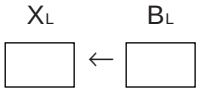
Operation data length: 16 bits or 8 bits

Operation : X ← B

When x = "0"



When x = "1"



❋ In this case, the contents of X_H do not change.

Description : Transfers the contents of B to X. The contents of B do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TBX	81_{16}, $C4_{16}$	2	2

Description example:

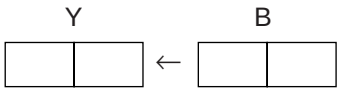
```
CLP   x
TBX                ; X ← B
SEP   x
TBX                ; XL ← BL
```

Function : Transfer between registers

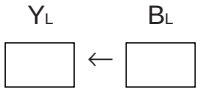
Operation data length: 16 bits or 8 bits

Operation : Y ← B

When x = "0"



When x = "1"



❋ In this case, the contents of Y_H do not change.

Description : Transfers Y with the contents of B. The contents of B do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TBY	81_{16}, $D4_{16}$	2	2

Description example:

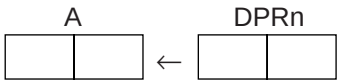
```
CLP   x
TBY                ; Y ← B
SEP   x
TBY                ; YL ← BL
```

Function : Transfer between registers

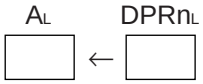
Operation data length: 16 bits or 8 bits

Operation : A ← DPRn (n = 0 to 3)

When m = "0"



When m = "1"



❋ In this case, the contents of A_H do not change.

Description : Transfers the contents of the specified DPRn (DPR0 to DPR3) to A.

- Specify one of DPR0 to DPR3 for the destination of transfer.
- The contents of DPRn do not change.
- This instruction includes the function of the TDA instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TDA n	31_{16}, $n2_{16}$+40_{16}	2	2

Note : Any value from 0 to 3 can be set to n.

Description example:

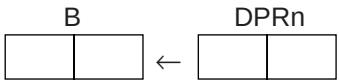
```
TDA   0     ; A ← DPR0
TDA   1     ; A ← DPR1
```

Function : Transfer between registers

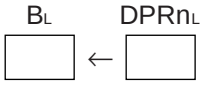
Operation data length: 16 bits or 8 bits

Operation : B ← DPRn (n = 0 to 3)

When m = "0"



When m = "1"



❋ In this case, the contents of B_H do not change.

Description : Transfers the contents of specified DPRn (DPR0 to DPR3) to B.

- Specify one of DPR0 to DPR3 for the destination of transfer.
- The contents of DPRn do not change.
- This instruction includes the function of the TDB instruction in the conventional 7700 Family.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TDB n	$B1_{16}$, $n2_{16}+40_{16}$	2	2

Note : Any value from 0 to 3 can be set to n.

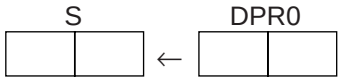
Description example:

```
TDB   0     ; B ← DPR0
TDB   1     ; B ← DPR1
```

Function : Transfer between registers

Operation data length: 16 bits

Operation : S ← DPR0



Description : Transfers the contents of DPR0 to S in 16-bit length.
- The contents of DPR0 do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TDS	31_{16}, 73_{16}	2	2

Description example:

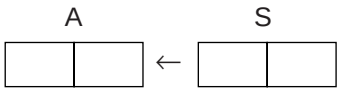
```
TDS                                    ; S ← DPR0
```

Function : Transfer between registers

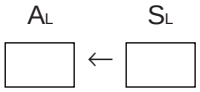
Operation data length: 16 bits or 8 bits

Operation : A ← S

When m = "0"



When m = "1"



❋ The contents of A_H do not change.

Description : Transfers the contents of S to A. The contents of S do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TSA	31_{16}, 92_{16}	2	2

Description example:

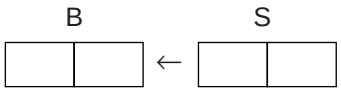
```
CLM
TSA                    ; A ← S
SEM
TSA                    ; AL ← SL
```

Function : Transfer between registers

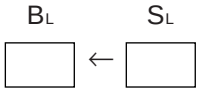
Operation data length: 16 bits or 8 bits

Operation : B ← S

When m = "0"



When m = "1"



❋ The contents of B_H do not change.

Description : Transfers the contents of S to B. The contents of S do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TSB	$B1_{16}$, 92_{16}	2	2

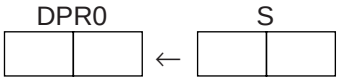
Description example:

```
CLM
TSB                    ; B ← S
SEM
TSB                    ; BL ← SL
```

Function : Transfer between registers

Operation data length: 16 bits

Operation : DPR0 ← S



Description : Transfers the contents of S to DPR0 in 16-bit length.
- The contents of S do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TSD	31_{16}, 70_{16}	2	4

Description example:

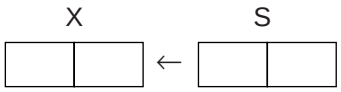
```
TSD                    ; DPR0 ← S
```

Function : Transfer between registers

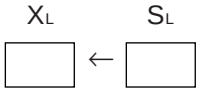
Operation data length: 16 bits or 8 bits

Operation : X ← S

When x = "0"



When x = "1"



❋ The contents of X_H do not change.

Description : Transfers the contents of S to X. The contents of S do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TSX	31_{16}, $F2_{16}$	2	2

Description example:

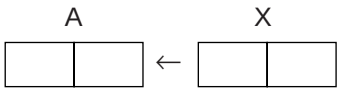
```
CLP     x
TSX                     ; X ← S
SEP     x
TSX                     ; XL ← SL
```

Function : Transfer between registers

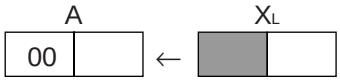
Operation data length: 16 bits or 8 bits

Operation : A ← X

When m = “0” and x = “0”

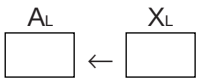


When m = “0” and x = “1”



❋ The data “00_{16}” is set to A_H.

When m = “1”



❋ The contents of A_H do not change.

Description : Transfers the contents of X to A. The contents of X do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TXA	$A4_{16}$	1	1

Description example:

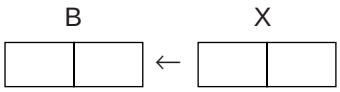
```
TXA                    ; A ← X
```

Function : Transfer between registers

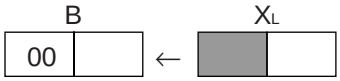
Operation data length: 16 bits or 8 bits

Operation : B ← X

When m = "0" and x = "0"

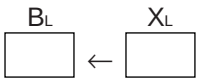


When m = "0" and x = "1"



❋ The data "00_{16}" is set to B_H.

When m = "1"



❋ The contents of B_H do not change.

Description : Transfers the contents of X to B. The contents of X do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TXB	81_{16}, $A4_{16}$	2	2

Description example:

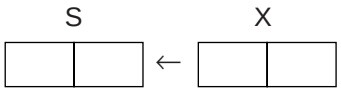
```
TXB                                    ; B ← X
```

Function : Transfer between registers

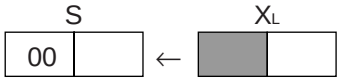
Operation data length: 16 bits or 8 bits

Operation : S ← X

When x = "0"



When x = "1"



❋ The data "00_{16}" is set to S_H.

Description : Transfers the contents of X to S. The contents of X do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TXS	31_{16}, $E2_{16}$	2	2

Description example:

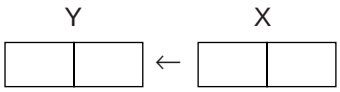
```
CLP     x
TXS                     ; S ← X
SEP     x
TXS                     ; SL ← XL, SH ← 00₁₆
```

Function : Transfer between registers

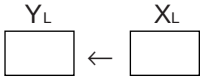
Operation data length: 16 bits or 8 bits

Operation : $Y \leftarrow X$

When x = "0"



When x = "1"



❋ The contents of Y_H do not change.

Description : Transfers the contents of X to Y. The contents of X do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TXY	31_{16}, $C2_{16}$	2	2

Description example:

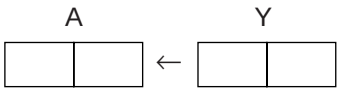
```
CLP     x
TXY                     ; Y ← X
SEP     x
TXY                     ; YL ← XL
```

Function : Transfer between registers

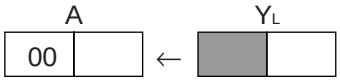
Operation data length: 16 bits or 8 bits

Operation : A ← Y

When m = "0" and x = "0"

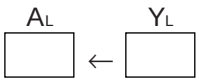


When m = "0" and x = "1"



❋ The data "00_{16}" is set to A_H.

When m = "1"



❋ The contents of A_H do not change.

Description : Transfers the contents of Y to A. The contents of Y do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TYA	$B4_{16}$	1	1

Description example:

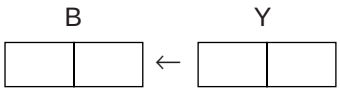
```
TYA                    ; A ← Y
```

Function : Transfer between registers

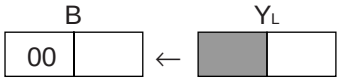
Operation data length: 16 bits or 8 bits

Operation : B ← Y

When m = "0" and x = "0"

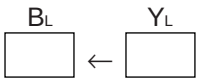


When m = "0" and x = "1"



❋ The data "00_{16}" is set to B_H.

When m = "1"



❋ The contents of B_H do not change.

Description : Transfers the contents of Y to B. The contents of Y do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TYB	81_{16}, $B4_{16}$	2	2

Description example:

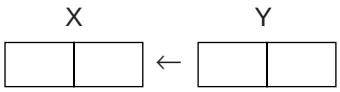
```
TYB                                    ; B ← Y
```

Function : Transfer between registers

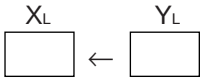
Operation data length: 16 bits or 8 bits

Operation : X ← Y

When x = "0"



When x = "1"



❋ The contents of X_H do not change.

Description : Transfers the contents of Y to X. The contents of Y do not change.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to "1" when MSB of the operation result is "1." Otherwise, cleared to "0."

Z : Set to "1" when the operation result is "0." Otherwise, cleared to "0."

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	TYX	31_{16}, $D2_{16}$	2	2

Description example:

```
CLP     x
TYX                     ; X ← Y
SEP     x
TYX                     ; XL ← YL
```

Function : Clock control

Operation data length: –

Operation : Stop the CPU clock.

Description : Stops the internal clock. However, the oscillation of the oscillation circuit is not stopped. To restart the internal clock, generate an interrupt request or perform the hardware reset. The microcomputer will thereby be released from the WIT state.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	—	—	—	—	—	—	—	—

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	WIT	31_{16}, 10_{16}	2	–

Description example:

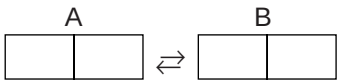
```
WIT                                    ;
```

Function : Transfer between registers

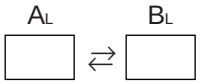
Operation data length: 16 bits or 8 bits

Operation : A ⇄ B

When m = “0”



When m = “1”



❋ In this case, the contents of A_H and B_H do not change.

Description : Exchanges the contentss of A and B.

Status flags :

IPL	N	V	m	x	D	I	Z	C
—	N	—	—	—	—	—	Z	—

N : Set to “1” when MSB of the operation result is “1.” Otherwise, cleared to “0.”

Z : Set to “1” when the operation result is “0.” Otherwise, cleared to “0.”

Addressing mode	Syntax	Machine code	Bytes	Cycles
IMP	XAB	55_{16}	1	2

Description example:

```
CLM     x
XAB                     ; A ⇄ B
SEM     x
XAB                     ; AL ⇄ BL
```

4.3 Notes on software development

The following are notes on software development.

4.3.1 Instruction execution cycles

The number of instruction execution cycles shown in this manual is applied to an ideal operating state. The actual instruction execution cycles vary with the instruction queue, the bus width for memory access, and the setting for Wait state.

When estimating a theoretical program execution speed by using the values shown in this manual or when implementing timers by software, be sure to consider that the estimated or anticipated execution time is only an approximate value.

4.3.2 Status of flags m and x

Writing a 16-bit immediate value to the instruction operand while the contents of flag m is "1" (8 bits of data length) or an 8-bit immediate value to the instruction operand while the contents of flag m is "0" (16 bits of data length) causes the program to run out of control.

The above is also applied to flag x. Refer to the user's manual of the assembler you are using and make sure that no discrepancy will occur between the flag state and the data length to be operated on.

4.3.3 Tips for data area location

(1) If the contents of low-order 8 bits of the direct page register (DPR_{nL}) are set to any value other than "00_{16}," the processing time is extended by 1 machine cycle as compared to the cases where the contents are set to "00_{16}." Therefore, Mitsubishi recommends setting these low-order bits to "00_{16}" whenever possible because this helps to increase the execution speed of program.

(2) Mitsubishi recommends locating 16-bit data at even address boundaries whenever possible because this is effective for increasing the program execution speed. If 16-bit data are located at odd address boundaries, 2 bus cycles need to be generated for accessing this data, resulting in a reduced program execution speed.

4.3.4 Performing arithmetic operations in decimal

(1) Arithmetic operations can be performed in decimal by setting flag D to "1." However, decimal operations can be performed only by the following 4 instructions:

- ADC
- ADCB
- SBC
- SBCB

(2) Pay attention to the flag behavior when performing decimal operations. Although the results of decimal operations are reflected correctly in flag C, the results are not reflected in any of flags Z, N, and V.

APPENDIX

Appendix 1. 7900 Series machine instructions

[How to use this table]

- The corresponding op code, the number of execution cycles, and the number of instruction bytes are indicated for each addressing mode of each instruction.
- A flag affected by the operation result is also indicated.
- For symbols used in this table, refer to the table on the next page. Also, refer to **"Notes for machine instruction table"** on pages 5-42 and 5-43.
- The operation length of an instruction of which column "Operation length (Bit)" includes "16/8" depends on the setting of flag m or x.

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Description	Symbol	Description
IMP	Implied addressing mode	E	Accumulator E
IMM	Immediate addressing mode	E_H	Accumulator E's high-order 16 bits (Accumulator B)
A	Accumulator addressing mode	E_L	Accumulator E's low-order 16 bits (Accumulator A)
DIR	Direct addressing mode	X	Index register X
DIR, X	Direct indexed X addressing mode	X_H	Index register X's high-order 8 bits
DIR, Y	Direct indexed Y addressing mode	X_L	Index register X's low-order 8 bits
(DIR)	Direct indirect addressing mode	Y	Index register Y
(DIR, X)	Direct indexed X indirect addressing mode	Y_H	Index register Y's high-order 8 bits
(DIR), Y	Direct indirect indexed Y addressing mode	Y_L	Index register Y's low-order 8 bits
L(DIR)	Direct indirect long addressing mode	S	Stack pointer
L(DIR), Y	Direct indirect long indexed Y addressing mode	REL	Relative address
ABS	Absolute addressing mode	PC	Program counter
ABS, X	Absolute indexed X addressing mode	PC_H	Program counter's high-order 8 bits
ABS, Y	Absolute indexed Y addressing mode	PC_L	Program counter's low-order 8 bits
ABL	Absolute long addressing mode	PG	Program bank register
ABL, X	Absolute long indexed X addressing mode	DT	Data back register
(ABS)	Absolute indirect addressing mode	DPR0	Direct page register 0
L(ABS)	Absolute indirect long addressing mode	$DPR0_H$	Direct page register 0's high-order 8 bits
(ABS, X)	Absolute indexed X indirect addressing mode	$DPR0_L$	Direct page register 0's low-order 8 bits
STK	Stack addressing mode	DPRn	Direct page register n
REL	Relative addressing mode	$DPRn_H$	Direct page register n's high-order 8 bits
DIR, b, R	Direct bit relative addressing mode	$DPRn_L$	Direct page register n's low-order 8 bits
ABS, b, R	Absolute bit relative addressing mode	PS	Processor status register
SR	Stack pointer relative addressing mode	PS_H	Processor status register's high-order 8 bits
(SR), Y	Stack pointer relative indirect indexed Y addressing mode	PS_L	Processor status register's low-order 8 bits
		PS_L(bit n)	nth bit in processor status register
BLK	Block transfer addressing mode	M	Contents of memory
Multiplied accumulation	Multiplied accumulation addressing mode	M(S)	Contents of memory at address indicated by stack pointer
op	Instruction code (Op code)	M(bit n)	nth bit of memory
n	Number of cycles	Mn	n-bit memory's address or contents
#	Number of bytes	IMM	Immediate value (8 bits or 16 bits)
C	Carry flag	IMMn	n-bit immediate value
Z	Zero flag	IMM_H	16-bit immediate value's high-order 8 bits
I	Interrupt disable flag	IMM_L	16-bit immediate value's low-order 8 bits
D	Decimal operation mode flag	AD_H	Value of 24-bit address's high-order 8 bits (A_{23}–A_{16})
x	Index register length selection flag	AD_M	Value of 24-bit address's middle-order 8 bits (A_{15}–A_8)
m	Data length selection flag	AD_L	Value of 24-bit address's low-order 8 bits (A_7–A_0)
V	Overflow flag	EAR	Effective address (16 bits)
N	Negative flag	EAR_H	Effective address's high-order 8 bits
IPL	Processor interrupt priority level	EAR_L	Effective address's low-order 8 bits
+	Addition	imm	8-bit immediate value
–	Subtraction	imm_n	n-bit immediate value
×	Multiplication	dd	Displacement for DPR (8 bits or 16 bits)
÷	Division	i	Number of transfer bytes, rotation or repeated operations
∧	Logical AND	i_1, i_2	Number of registers pushed or pulled
∨	Logical OR	source	Operand to specify transfer source
∀	Logical exclusive OR	dest	Operand to specify transfer destination
| |	Absolute value		
—	Negation		
→	Movement to the arrow direction		
←	Movement to the arrow direction		
⇔	Exchange		
Acc	Accumulator		
Acc_H	Accumulator's high-order 8 bits		
Acc_L	Accumulator's low-order 8 bits		
A	Accumulator A		
A_H	Accumulator A's high-order 8 bits		
A_L	Accumulator A's low-order 8 bits		
B	Accumulator B		
B_H	Accumulator B's high-order 8 bits		
B_L	Accumulator B's low-order 8 bits		

APPENDIX
Appendix 1. 7900 Series machine instructions

7900 Series Machine Instructions

Addressing Modes (each cell: op; n; #)

Symbol	Function	Operation length (Bit)	IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y
ABS **(Note 1)**	Acc← | Acc |	16/8			E1; 3; 1 81 E1; 4; 2								
ABSD	E← | E |	32			31 90; 5; 2								
ADC **(Notes 1 and 2)**	Acc←Acc + M + C	16/8		31 87; 3; 3 B1 87; 3; 3		21 8A; 5; 3 A1 8A; 7; 3	21 8B; 6; 3 A1 8B; 8; 3		21 80; 7; 3 A1 80; 9; 3	21 81; 8; 3 A1 81; 10; 3	21 88; 8; 3 A1 88; 10; 3	21 82; 9; 3 A1 82; 11; 3	21 89; 10; 3 A1 89; 12; 3
ADCB **(Note 1)**	AccL←AccL + IMM8 + C	8		31 1A; 3; 3 B1 1A; 3; 3									
ADCD	E←E + M32 + C	32		31 1C; 4; 6		21 9A; 7; 3	21 9B; 8; 3		21 90; 9; 3	21 91; 10; 3	21 98; 10; 3	21 92; 11; 3	21 99; 12; 3
ADD **(Notes 1 and 2)**	Acc←Acc + M	16/8		26; 1; 2 81 26; 2; 3		2A; 3; 2 81 2A; 4; 3	2B; 4; 2 81 2B; 5; 3		11 20; 6; 3 91 20; 6; 3	11 21; 7; 3 91 21; 7; 3	11 28; 7; 3 91 28; 7; 3	11 22; 8; 3 91 22; 8; 3	11 29; 9; 3 91 29; 9; 3
ADDB **(Note 1)**	AccL←AccL + IMM8	8		29; 1; 2 81 29; 2; 3									
ADDD	E←E + M32	32		2D; 3; 5		9A; 6; 2	9B; 7; 2		11 90; 9; 3	11 91; 10; 3	11 98; 10; 3	11 92; 11; 3	11 99; 12; 3
ADDM **(Note 3)**	M←M + IMM	16/8				51 03; 7; 4							
ADDMB	M8←M8 + IMM8	8				51 02; 7; 4							
ADDMD	M32←M32 + IMM32	32				51 83; 10; 7							
ADDS	S←S + IMM8	16		31 0A; 2; 3									
ADDX	X←X + IMM (IMM = 0 to 31)	16/8		01; 2; 2									
ADDY **(Note 4)**	Y←Y + IMM (IMM = 0 to 31)	16/8		01 20 + imm; 2; 2									

Addressing Modes (each cell: op; n; #) and Processor Status register (bits 10–8: IPL; 7: N; 6: V; 5: m; 4: x; 3: D; 2: I; 1: Z; 0: C)

Symbol	ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	IPL (10 9 8)	N (7)	V (6)	m (5)	x (4)	D (3)	I (2)	Z (1)	C (0)
ABS																	• • •	0	V	•	•	•	•	Z	0
ABSD																	• • •	0	V	•	•	•	•	Z	0
ADC	21 8E; 5; 4 A1 8E; 7; 4	21 8F; 6; 4 A1 8F; 8; 4	21 86; 6; 4 A1 86; 8; 4	21 8C; 6; 5 A1 8C; 8; 5	21 8D; 7; 5 A1 8D; 9; 5								21 83; 6; 3 A1 83; 8; 3	21 84; 9; 3 A1 84; 11; 3			• • •	N	V	•	•	•	•	Z	C
ADCB																	• • •	N	V	•	•	•	•	Z	C
ADCD	21 9E; 7; 4	21 9F; 8; 4	21 96; 8; 4	21 9C; 8; 5	21 9D; 9; 5								21 93; 8; 3	21 94; 11; 3			• • •	N	V	•	•	•	•	Z	C
ADD	2E; 3; 3 81 2E; 4; 4	2F; 4; 3 81 2F; 5; 4	11 26; 5; 4 91 26; 5; 4	11 2C; 5; 5 91 2C; 5; 5	11 2D; 6; 5 91 2D; 6; 5								11 93; 5; 3 91 23; 5; 3	11 24; 8; 3 91 24; 8; 3			• • •	N	V	•	•	•	•	Z	C
ADDB																	• • •	N	V	•	•	•	•	Z	C
ADDD	9E; 6; 3	9F; 7; 3	11 96; 8; 4	11 9C; 8; 5	11 9D; 9; 5								11 93; 8; 3	11 94; 11; 3			• • •	N	V	•	•	•	•	Z	C
ADDM	51 07; 7; 5																• • •	N	V	•	•	•	•	Z	C
ADDMB	51 06; 7; 5																• • •	N	V	•	•	•	•	Z	C
ADDMD	51 87; 10; 8																• • •	N	V	•	•	•	•	Z	C
ADDS																	• • •	N	V	•	•	•	•	Z	C
ADDX																	• • •	N	V	•	•	•	•	Z	C
ADDY																	• • •	N	V	•	•	•	•	Z	C

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes: IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y
			op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #
AND **(Notes 1 and 2)**	Acc←Acc∧M	16/8		66 1 2 81 66 2 3		6A 3 2 81 6A 4 3	6B 4 2 81 6B 5 3		11 60 6 3 91 60 6 3	11 61 7 3 91 61 7 3	11 68 7 3 91 68 7 3	11 62 8 3 91 62 8 3	11 69 9 3 91 69 9 3
ANDB **(Note 1)**	Acc_L←Acc_L∧IMM8	8		23 1 2 81 23 2 3									
ANDM **(Note 3)**	M←M∧IMM	16/8				51 63 7 4							
ANDMB	M8←M8∧IMM8	8				51 62 7 4							
ANDMD	M32←M32∧IMM32	32				51 E3 10 7							
ASL **(Note 1)**	Arithmetic shift to the left by 1 bit m = 0 Acc or M16 C←[b_{15}…b_0]←0 m = 1 Acc_L or M8 C←[b_7…b_0]←0	16/8			03 1 1 81 03 2 2	21 0A 7 3	21 0B 8 3						
ASL #n **(Note 4)**	Arithmetic shift to the left by n bits (n = 0 to 15) m = 0 A C←[b_{15}…b_0]←0 m = 1 A_L C←[b_7…b_0]←0	16/8			C1 40 6 + imm 2 + imm								
ASLD #n **(Note 4)**	Arithmetic shift to the left by n bits (n = 0 to 31) E C←[b_{31}…b_0]←0	32			D1 40 8 + imm 2 + imm								
ASR **(Note 1)**	Arithmetic shift to the right by 1 bit m = 0 Acc or M16 →[b_{15}…b_0]→C m = 1 Acc_L or M8 →[b_7…b_0]→C	16/8			64 1 1 81 64 2 2	21 4A 7 3	21 4B 8 3						
ASR #n **(Note 4)**	Arithmetic shift to the right by n bits (n = 0 to 15) m = 0 A →[b_{15}…b_0]→C m = 1 A_L →[b_7…b_0]→C	16/8			C1 80 6 + imm 2 + imm								

Addressing Modes: ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	Processor Status register: 10 9 8 7 6 5 4 3 2 1 0
op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	IPL N V m x D I Z C
6E 3 3	6F 4 3	11 66 5 4	11 6C 5 5	11 6D 6 5								11 63 5 3	11 64 8 3			• • • N • • • • • Z •
81 6E 4 4	81 6F 5 4	91 66 5 4	91 6C 5 5	91 6D 6 5								91 63 5 3	91 64 8 3			
51 67 7 5																• • • N • • • • • Z •
51 66 7 5																• • • N • • • • • Z •
51 E7 10 8																• • • N • • • • • Z •
21 0E 7 4	21 0F 8 4															• • • N • • • • • Z C
																• • • N • • • • • Z C
																• • • N • • • • • Z C
21 4E 7 4	21 4F 8 4															• • • N • • • • • Z C
																• • • N • • • • • Z C

APPENDIX
Appendix 1. 7900 Series machine instructions

Addressing Modes (each cell: op / n / #)

Symbol	Function	Operation length (Bit)	IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y
ASRD #n **(Note 4)**	Arithmetic shift to the right by n bits (n = 0 to 31) E → b31 … b0 → C	32			D1 80 + imm / 8 + imm / 2								
BBC **(Note 3)**	if M(bit n) = 0 then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8											
BBCB	if M8(bit n) = 0 then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	8											
BBS **(Note 3)**	if M(bit n) = 1 then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8											
BBSB	if M8(bit n) = 1 then PC←PC+cnt+REL (–128 to +127) (cnt: Number of bytes of instruction)	8											
BCC	if C = 0 then PC←PC + 2 + REL (–128 to +127)	–											
BCS	if C = 1 then PC←PC + 2 + REL (–128 to +127)	–											
BEQ	if Z = 1 then PC←PC + 2 + REL (–128 to +127)	–											
BGE	if N∀V = 0 then PC←PC + 2 + REL (–128 to +127)	–											
BGT	if Z = 0 and N∀V = 0 then PC←PC + 2 + REL (–128 to +127)	–											
BGTU	if C = 1 and Z = 0 then PC←PC + 2 + REL (–128 to +127)	–											
BLE	if Z = 1 or N∀V = 1 then PC←PC + 2 + REL (–128 to +127)	–											
BLEU	if C = 0 or Z = 1 then PC←PC + 2 + REL(–128 to +127)	–											
BLT	if N∀V = 1 then PC←PC + 2 + REL (–128 to +127)	–											

Addressing Modes (continued; each cell: op / n / #)

Symbol	ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA
ASRD #n																
BBC											41 5A / 9 / 5	41 5E / 9 / 6				
BBCB											52 / 8 / 4	57 / 8 / 5				
BBS											41 4A / 9 / 5	41 4E / 9 / 6				
BBSB											42 / 8 / 4	47 / 8 / 5				
BCC										90 / 6 / 2						
BCS										B0 / 6 / 2						
BEQ										F0 / 6 / 2						
BGE										C0 / 6 / 2						
BGT										80 / 6 / 2						
BGTU										40 / 6 / 2						
BLE										A0 / 6 / 2						
BLEU										60 / 6 / 2						
BLT										E0 / 6 / 2						

Processor Status register

Symbol	10 (IPL)	9 (IPL)	8 (IPL)	7 N	6 V	5 m	4 x	3 D	2 I	1 Z	0 C
ASRD #n	•	•	•	N	•	•	•	•	•	Z	C
BBC	•	•	•	•	•	•	•	•	•	•	•
BBCB	•	•	•	•	•	•	•	•	•	•	•
BBS	•	•	•	•	•	•	•	•	•	•	•
BBSB	•	•	•	•	•	•	•	•	•	•	•
BCC	•	•	•	•	•	•	•	•	•	•	•
BCS	•	•	•	•	•	•	•	•	•	•	•
BEQ	•	•	•	•	•	•	•	•	•	•	•
BGE	•	•	•	•	•	•	•	•	•	•	•
BGT	•	•	•	•	•	•	•	•	•	•	•
BGTU	•	•	•	•	•	•	•	•	•	•	•
BLE	•	•	•	•	•	•	•	•	•	•	•
BLEU	•	•	•	•	•	•	•	•	•	•	•
BLT	•	•	•	•	•	•	•	•	•	•	•

Symbol	Function	Operation length (Bit)	IMP op	IMP n	IMP #	IMM op	IMM n	IMM #	A op	A n	A #	DIR op	DIR n	DIR #	DIR, X op	DIR, X n	DIR, X #	DIR, Y op	DIR, Y n	DIR, Y #	(DIR) op	(DIR) n	(DIR) #	(DIR, X) op	(DIR, X) n	(DIR, X) #	(DIR), Y op	(DIR), Y n	(DIR), Y #	L(DIR) op	L(DIR) n	L(DIR) #	L(DIR), Y op	L(DIR), Y n	L(DIR), Y #
BMI	if N = 1 then PC←PC + 2 + REL (–128 to +127)	–																																	
BNE	if Z = 0 then PC←PC + 2 + REL (–128 to +127)	–																																	
BPL	if N = 0 then PC←PC + 2 + REL (–128 to +127)	–																																	
BRA/BRAL **(Note 5)**	PC←PC + cnt + REL (BRA:–128 to +127, BRAL: –32768 to +32767) (cnt: Number of bytes of instruction) PG←PG + 1 (When carry occurs) PG←PG – 1 (When borrow occurs)	–																																	
BRK **(Note 6)**	PC←PC + 2 M(S)←PG S←S – 1 M(S)←PC_H S←S – 1 M(S)←PC_L S←S – 1 M(S)←PS_H S←S – 1 M(S)←PS_L S←S – 1 I←1 PC_L←AD_L PC_H←AD_M PG←00_{16} or FF_{16}	–	00 74	15	2																														
BSC **(Note 7)**	if A(bit n) or M(bit n) = 0 (n = 0 to 15), then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8							01 A0 + n	7	3	71 A0 + n	11	4																					
BSR	(S)←PC PC←PC + 2 + REL (–1024 to +1023)	–																																	
BSS **(Note 7)**	if A(bit n) or M(bit n) = 1 (n = 0 to 15), then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8							01 80 + n	7	3	71 80 + n	11	4																					
BVC	if V = 0 then PC←PC + 2 + REL (–128 to +127)	–																																	
BVS	if V = 1 then PC←PC + 2 + REL (–128 to +127)	–																																	

Symbol	ABS op	ABS n	ABS #	ABS, X op	ABS, X n	ABS, X #	ABS, Y op	ABS, Y n	ABS, Y #	ABL op	ABL n	ABL #	ABL, X op	ABL, X n	ABL, X #	(ABS) op	(ABS) n	(ABS) #	L(ABS) op	L(ABS) n	L(ABS) #	(ABS, X) op	(ABS, X) n	(ABS, X) #	STK op	STK n	STK #
BMI																											
BNE																											
BPL																											
BRA/BRAL																											
BRK																											
BSC	71 E + n	10	5																								
BSR																											
BSS	71 C0 + n	10	5																								
BVC																											
BVS																											

Symbol	REL op	REL n	REL #	DIR, b, R op	DIR, b, R n	DIR, b, R #	ABS, b, R op	ABS, b, R n	ABS, b, R #	SR op	SR n	SR #	(SR), Y op	(SR), Y n	(SR), Y #	BLK op	BLK n	BLK #	MAA op	MAA n	MAA #	10 IPL	9 IPL	8 IPL	7 N	6 V	5 m	4 x	3 D	2 I	1 Z	0 C
BMI	30	6	2																			•	•	•	•	•	•	•	•	•	•	•
BNE	D0	6	2																			•	•	•	•	•	•	•	•	•	•	•
BPL	10	6	2																			•	•	•	•	•	•	•	•	•	•	•
BRA/BRAL	20 A7	5 5	2 3																			•	•	•	•	•	•	•	•	•	•	•
BRK																						•	•	•	•	•	•	•	•	1	•	•
BSC																						•	•	•	•	•	•	•	•	•	•	•
BSR	F8–FF	7	2																			•	•	•	•	•	•	•	•	•	•	•
BSS																						•	•	•	•	•	•	•	•	•	•	•
BVC	50	6	2																			•	•	•	•	•	•	•	•	•	•	•
BVS	70	6	2																			•	•	•	•	•	•	•	•	•	•	•

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y	ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	10	9	8	7	6	5	4	3	2	1	0
			op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	IPL	IPL	IPL	N	V	m	x	D	I	Z	C
CBEQ **(Notes 1 and 3)**	if Acc = IMM or M = IMM then PC←PC + cnt + REL(–128 to +127) (cnt: Number of bytes of instruction)	16/8			A6 6 3 81 A6 7 4	41 6A 9 5																								•	•	•	N	V	•	•	•	•	Z	C
CBEQB **(Note 1)**	if Acc$_L$ = IMM8 or M8 = IMM8 then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	8			A2 6 3 81 A2 7 4	62 8 4																								•	•	•	N	V	•	•	•	•	Z	C
CBNE **(Notes 1 and 3)**	if Acc ≠ IMM or M ≠ IMM then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8			B6 6 3 81 B6 7 4	41 7A 9 5																								•	•	•	N	V	•	•	•	•	Z	C
CBNEB **(Note 1)**	if Acc$_L$ ≠ IMM8 or M8 ≠ IMM8 then PC←PC+cnt+REL(–128 to +127) (cnt: Number of bytes of instruction)	8			B2 6 3 81 B2 7 4	72 8 4																								•	•	•	N	V	•	•	•	•	Z	C
CLC	C←0	–	14 1 1																											•	•	•	•	•	•	•	•	•	•	0
CLI	I←0	–	15 3 1																											•	•	•	•	•	•	•	•	0	•	•
CLM	m←0	–	45 3 1																											•	•	•	•	•	0	•	•	•	•	•
CLP	PS$_L$(bit n)←0 (n = 0 to 7. Multiple bits can be specified.)	–		98 4 2																										•	•	•	Specified flag becomes "0."							
CLR **(Note 1)**	Acc←0	16/8			54 1 1 81 54 2 2																									•	•	•	0	•	•	•	•	•	1	•
CLRB **(Note 1)**	Acc$_L$←00$_{16}$	8			44 1 1 81 44 2 2																									•	•	•	0	•	•	•	•	•	1	•
CLRM	M←0	16/8				D2 5 2								D7 5 3																•	•	•	•	•	•	•	•	•	•	•
CLRMB	M8←00$_{16}$	8				C2 5 2								C7 5 3																•	•	•	•	•	•	•	•	•	•	•
CLRX	X←0	16/8	E4 1 1																											•	•	•	0	•	•	•	•	•	1	•
CLRY	Y←0	16/8	F4 1 1																											•	•	•	0	•	•	•	•	•	1	•

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	IMP			IMM			A			DIR			DIR, X		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
CLV	V←0	–	65	1	1												
CMP **(Notes 1 and 2)**	Acc – M	16/8				46	1	2				4A	3	2	4B	4	2
						81 46	2	3				81 4A	4	3	81 4B	5	3
CMPB **(Note 1)**	Acc$_L$ – IMM8	8				38	1	2									
						81 38	2	3									
CMPD	E – IMM32	32				3C	3	5				BA	6	2	BB	7	2
CMPM **(Note 3)**	M – IMM	16/8										51 23	5	4			
CMPMB	M8 – IMM8	8										51 22	5	4			
CMPMD	M32 – IMM32	32										51 A3	7	7			
CPX **(Note 8)**	X – M	16/8				E6	1	2				22	3	2			
CPY **(Note 8)**	Y – M	16/8				F6	1	2				32	3	2			
DEBNE **(Note 4)**	M←M – IMM(IMM = 0 to 31) if M ≠ 0, then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8										C1 A0 + imm	12	4			
DEC **(Note 1)**	Acc←Acc – 1 or M←M – 1	16/8							B3	1	1	92	6	2	41 9B	8	3
									81 B3	2	2						
DEX	X←X – 1	16/8	E3	1	1												
DEY	Y←Y – 1	16/8	F3	1	1												
DIV **(Notes 2, 9, and 10)**	A (quotient) ← (B, A) ÷ M B (remainder)	16/8				31 E7	15	3				21 EA	16	3	21 EB	17	3

Symbol	DIR, Y			(DIR)			(DIR, X)			(DIR), Y			L(DIR)			L(DIR), Y		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
CLV																		
CMP				11 40	6	3	11 41	7	3	11 48	7	3	11 42	8	3	11 49	9	3
				91 40	6	3	91 41	7	3	91 48	7	3	91 42	8	3	91 49	9	3
CMPB																		
CMPD				11 B0	9	3	11 B1	10	3	11 B8	10	3	11 B2	11	3	11 B9	12	3
CMPM																		
CMPMB																		
CMPMD																		
CPX																		
CPY																		
DEBNE																		
DEC																		
DEX																		
DEY																		
DIV				21 E0	18	3	21 E1	19	3	21 E8	19	3	21 E2	20	3	21 E9	21	3

Symbol	ABS			ABS, X			ABS, Y			ABL			ABL, X			(ABS)			L(ABS)			(ABS, X)		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
CLV																								
CMP	4E	3	3	4F	4	3	11 46	5	4	11 4C	5	5	11 4D	6	5									
	81 4E	4	4	81 4F	5	4	91 46	5	4	91 4C	5	5	91 4D	6	5									
CMPB																								
CMPD	BE	6	3	BF	7	3	11 B6	8	4	11 BC	8	5	11 BD	9	5									
CMPM	51 27	5	5																					
CMPMB	51 26	5	5																					
CMPMD	51 A7	7	8																					
CPX	41 2E	4	4																					
CPY	41 3E	4	4																					
DEBNE	D1 E0 + imm	11	5																					
DEC	97	6	3	41 9F	8	4																		
DEX																								
DEY																								
DIV	21 EE	16	4	21 EF	17	4	21 E6	17	4	21 EC	17	5	21 ED	18	5									

Symbol	STK			REL			DIR, b, R			ABS, b, R			SR			(SR), Y			BLK			MAA		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
CLV																								
CMP													11 43	5	3	11 44	8	3						
													91 43	5	3	91 44	8	3						
CMPB																								
CMPD													11 B3	8	3	11 B4	11	3						
CMPM																								
CMPMB																								
CMPMD																								
CPX																								
CPY																								
DEBNE																								
DEC																								
DEX																								
DEY																								
DIV													21 E3	17	3	21 E4	20	3						

Symbol	Processor Status register										
	10	9	8	7	6	5	4	3	2	1	0
	IPL			N	V	m	x	D	I	Z	C
CLV	•	•	•	•	0	•	•	•	•	•	•
CMP	•	•	•	N	V	•	•	•	•	Z	C
CMPB	•	•	•	N	V	•	•	•	•	Z	C
CMPD	•	•	•	N	V	•	•	•	•	Z	C
CMPM	•	•	•	N	V	•	•	•	•	Z	C
CMPMB	•	•	•	N	V	•	•	•	•	Z	C
CMPMD	•	•	•	N	V	•	•	•	•	Z	C
CPX	•	•	•	N	V	•	•	•	•	Z	C
CPY	•	•	•	N	V	•	•	•	•	Z	C
DEBNE	•	•	•	•	•	•	•	•	•	•	•
DEC	•	•	•	N	•	•	•	•	•	Z	•
DEX	•	•	•	N	•	•	•	•	•	Z	•
DEY	•	•	•	N	•	•	•	•	•	Z	•
DIV	•	•	•	N	V	•	•	•	I	Z	C

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes: IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y
DIVS **(Notes 2, 9, and 10)**	A (quotient) ←(B, A) ÷ M B (remainder) (Signed)	16/8		31 F7 22 3		21 FA 23 3	21 FB 24 3		21 F0 25 3	21 F1 26 3	21 F8 26 3	21 F2 27 3	21 F9 28 3
DXBNE **(Note 4)**	X←X – IMM (IMM = 0 to 31) if X ≠ 0, then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8		01 C0 + imm 7 3									
DYBNE **(Note 4)**	Y←Y – IMM (IMM = 0 to 31) if Y≠0, then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8		01 E0 + imm 7 3									
EOR **(Notes 1 and 2)**	Acc←Acc∀M	16/8		76 1 2 81 76 2 3		7A 3 2 81 7A 4 3	7B 4 2 81 7B 5 3		11 70 6 3 91 70 6 3	11 71 7 3 91 71 7 3	11 78 7 3 91 78 7 3	11 72 8 3 91 72 8 3	11 79 9 3 91 79 9 3
EORB **(Note 1)**	Acc$_L$←Acc$_L$∀IMMB	8		33 1 2 81 33 2 3									
EORM **(Note 3)**	M←M∀IMM	16/8				51 73 7 4							
EORMB	M8←M8∀IMM8	8				51 72 7 4							
EORMD	M32←M32∀IMM32	32				51 F3 10 7							
EXTS **(Note 1)**	Acc←Acc$_L$ (Extension sign) (Bit 7 of Acc$_L$ = 0) b15 b7 b0 00000000 | 0 Acc$_H$ Acc$_L$ (Bit 7 of Acc$_L$ = 1) b15 b7 b0 11111111 | 1 Acc$_H$ Acc$_L$	16			35 1 1 81 35 2 2								
EXTSD	E←E$_L$(= A) (Extension sign) (Bit 15 of A = 0) b15 b0 b15 b0 00000000 | 0 E$_H$(B) E$_L$(A) (Bit 15 of A = 1) b15 b0 b15 b0 11111111 | 1 E$_H$(B) E$_L$(A)	32			31 B0 5 2								
EXTZ **(Note 1)**	Acc←Acc$_L$ (Extension zero) b15 b8 b7 b0 00000000 | Acc$_H$ Acc$_L$	16			34 1 1 81 34 2 2								
EXTZD	E←E$_L$(= A) (Extension zero) b15 b8 b7 b0 00000000 | E$_H$(B) E$_L$(A)	32			31 A0 3 2								

Symbol	Function	Operation length (Bit)	Addressing Modes: ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	Processor Status register: 10 9 8 7 6 5 4 3 2 1 0
																			IPL N V m x D I Z C
DIVS **(Notes 2, 9, and 10)**	A (quotient) ←(B, A) ÷ M B (remainder) (Signed)	16/8	21 FE 23 4	21 FF 24 4	21 F6 24 4	21 FC 24 5	21 FD 25 5								21 F3 24 3	21 F4 27 3			• • • N V • • • I Z C
DXBNE **(Note 4)**	X←X – IMM (IMM = 0 to 31) if X ≠ 0, then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8																	• • • • • • • • • • •
DYBNE **(Note 4)**	Y←Y – IMM (IMM = 0 to 31) if Y≠0, then PC←PC + cnt + REL (–128 to +127) (cnt: Number of bytes of instruction)	16/8																	• • • • • • • • • • •
EOR **(Notes 1 and 2)**	Acc←Acc∀M	16/8	7E 3 3 81 7E 4 4	7F 4 3 81 7F 5 4	11 76 5 4 91 76 5 4	11 7C 5 5 91 7C 5 5	11 7D 6 5 91 7D 6 5								11 73 5 3 91 73 5 3	11 74 8 3 91 74 8 3			• • • N • • • • • Z •
EORB **(Note 1)**	Acc$_L$←Acc$_L$∀IMMB	8																	• • • N • • • • • Z •
EORM **(Note 3)**	M←M∀IMM	16/8	51 77 7 5																• • • N • • • • • Z •
EORMB	M8←M8∀IMM8	8	51 76 7 5																• • • N • • • • • Z •
EORMD	M32←M32∀IMM32	32	51 F7 10 8																• • • N • • • • • Z •
EXTS **(Note 1)**	Acc←Acc$_L$ (Extension sign) (Bit 7 of Acc$_L$ = 0) b15 b7 b0 00000000 | 0 Acc$_H$ Acc$_L$ (Bit 7 of Acc$_L$ = 1) b15 b7 b0 11111111 | 1 Acc$_H$ Acc$_L$	16																	• • • N • • • • • Z •
EXTSD	E←E$_L$(= A) (Extension sign) (Bit 15 of A = 0) b15 b0 b15 b0 00000000 | 0 E$_H$(B) E$_L$(A) (Bit 15 of A = 1) b15 b0 b15 b0 11111111 | 1 E$_H$(B) E$_L$(A)	32																	• • • N • • • • • Z •
EXTZ **(Note 1)**	Acc←Acc$_L$ (Extension zero) b15 b8 b7 b0 00000000 | Acc$_H$ Acc$_L$	16																	• • • 0 • • • • • Z •
EXTZD	E←E$_L$(= A) (Extension zero) b15 b8 b7 b0 00000000 | E$_H$(B) E$_L$(A)	32																	• • • 0 • • • • • Z •

Symbol	Function	Operation length (Bit)	IMP			IMM			A			DIR			DIR, X			DIR, Y			(DIR)			(DIR, X)			(DIR), Y			L(DIR)			L(DIR), Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
INC **(Note 1)**	Acc←Acc + 1 or M←M + 1	16/8							A3	1	1	82	6	2	41 8B	8	3																		
									81 A3	2	2																								
INX	X←X + 1	16/8	C3	1	1																														
INY	Y←Y + 1	16/8	D3	1	1																														
JMP/JMPL	When ABS specified PC_L←AD_L PC_H←AD_M When ABL specified PC_L←AD_L PC_H←AD_M PG←AD_H When (ABS) specified PC_L←(AD_M, AD_L) PC_H←(AD_M, AD_L + 1) When L(ABS) specified PC_L←(AD_M, AD_L) PC_H←(AD_M, AD_L + 1) PG←(AD_M, AD_L + 2) When (ABS,X) specified PC_L←(AD_M, AD_L + X) PC_H←(AD_M, AD_L + X + 1)	–																																	
JSR/JSRL	When ABS specified M(S)←PC_H S←S–1 M(S)←PC_L S←S–1 PC_L←AD_L PC_H←AD_M When ABL specified M(S)←PG S←S – 1 M(S)←PC_H S←S – 1 M(S)←PC_L S←S – 1 PC_L←AD_L PC_H←AD_M PG←AD_H When (ABS,X) specified M(S)←PC_H S←S – 1 M(S)←PC_L S←S – 1 PC_L←(AD_M, AD_L + X) PC_H←(AD_M, AD_L + X + 1)	–																																	
LDA **(Notes 1 and 2)**	Acc←M	16/8				16	1	2				1A	3	2	1B	4	2				11 10	6	3	11 11	7	3	18	6	2	11 12	8	3	19	8	2
						81 16	2	3				81 1A	4	3	81 1B	5	3				91 10	6	3	91 11	7	3	81 18	7	3	91 12	8	3	81 19	9	3
LDAB **(Note 1)**	Acc←M8 (Extension zero)	16				28	1	2				0A	3	2	0B	4	2				11 00	6	3	11 01	7	3	08	6	2	11 02	8	3	09	8	2
						81 28	2	3				81 0A	4	3	81 0B	5	3				91 00	6	3	91 01	7	3	81 08	7	3	91 02	8	3	81 09	9	3

Symbol	ABS			ABS, X			ABS, Y			ABL			ABL, X			(ABS)			L(ABS)			(ABS, X)		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
INC	87	6	3	41 8F	8	4																		
INX																								
INY																								
JMP/JMPL	9C	4	3							AC	5	4				31 5C	7	4	31 5D	9	4	BC	7	3
JSR/JSRL	9D	6	3							AD	7	4										BD	8	3
LDA	1E	3	3	1F	4	3	11 16	5	4	1C	4	4	1D	5	4									
	81 1E	4	4	81 1F	5	4	91 16	5	4	81 1C	5	5	81 1D	6	5									
LDAB	0E	3	3	0F	4	3	11 06	5	4	0C	4	4	0D	5	4									
	81 0E	4	4	81 0F	5	4	91 06	5	4	81 0C	5	5	81 0D	6	5									

Symbol	STK			REL			DIR, b, R			ABS, b, R			SR			(SR), Y			BLK			MAA			Processor Status register										
																									10	9	8	7	6	5	4	3	2	1	0
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
INC																									•	•	•	N	•	•	•	•	•	Z	•
INX																									•	•	•	N	•	•	•	•	•	Z	•
INY																									•	•	•	N	•	•	•	•	•	Z	•
JMP/JMPL																									•	•	•	•	•	•	•	•	•	•	•
JSR/JSRL																									•	•	•	•	•	•	•	•	•	•	•
LDA													11 13	5	3	11 14	8	3							•	•	•	N	•	•	•	•	•	Z	•
													91 13	5	3	91 14	8	3																	
LDAB													11 03	5	3	11 04	8	3							•	•	•	0	•	•	•	•	•	Z	•
													91 03	5	3	91 04	8	3																	

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)
LDAD	E←M32	32
LDD n **(Notes 11 and 12)**	DPRn←IMM16 (n = 0 to 3. Multiple DPRs can be specified.)	16
LDT	DT←IMM8	8
LDX **(Note 8)**	X←M	16/8
LDXB	X←IMM8 (Extension zero)	16
LDY **(Note 8)**	Y←M	16/8
LDYB	Y←IMM8 (Extension zero)	16
LSR **(Note 1)**	Logical shift to the right by 1 bit m = 0 Acc or M16 0→b15…b0→C m = 1 AccL or M8 0→b7…b0→C	16/8
LSR #n **(Note 4)**	Logical shift to the right by n bits (n = 0 to 15) m = 0 A 0→b15…b0→C m = 1 AL 0→b7…b0→C	16/8
LSRD #n **(Note 4)**	Logical shift to the right by n bits (n = 0 to 31) E 0→b31…b0→C	32

Addressing Modes

Symbol	IMP			IMM			A			DIR			DIR, X			DIR, Y			(DIR)			(DIR, X)			(DIR), Y			L(DIR)			L(DIR), Y		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
LDAD				2C	3	5				8A	6	2	8B	7	2				11 80	9	3	11 81	10	3	88	9	2	11 82	11	3	89	11	2
LDD n				B8 ?0 B8 ?0	13 11 + 2 i	4 2 + 2 i																											
LDT				31 4A	4	3																											
LDX				C6	1	2				02	3	2				41 05	5	3															
LDXB				27	1	2																											
LDY				D6	1	2				12	3	2	41 1B	5	3																		
LDYB				37	1	2																											
LSR							43 81 43	1 2	1 2	21 2A	7	3	21 2B	8	3																		
LSR #n							C1	6 + imm	2																								
LSRD #n							D1	8 + imm	2																								

Addressing Modes

Symbol	ABS			ABS, X			ABS, Y			ABL			ABL, X			(ABS)			L(ABS)			(ABS, X)			STK			REL			DIR, b, R			ABS, b, R			SR			(SR), Y			BLK			MAA		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
LDAD	8E	6	3	8F	7	3	11 86	8	4	8C	7	4	8D	8	4																						11 83	8	3	11 84	11	3						
LDD n																																																
LDT																																																
LDX	07	3	3				41 06	5	4																																							
LDXB																																																
LDY	17	3	3	41 1F	5	4																																										
LDYB																																																
LSR	21 2E	7	4	21 2F	8	4																																										
LSR #n																																																
LSRD #n																																																

Processor Status register

Symbol	10	9	8	7	6	5	4	3	2	1	0
	IPL			N	V	m	x	D	I	Z	C
LDAD	•	•	•	N	•	•	•	•	•	Z	•
LDD n	•	•	•	•	•	•	•	•	•	•	•
LDT	•	•	•	•	•	•	•	•	•	•	•
LDX	•	•	•	N	•	•	•	•	•	Z	•
LDXB	•	•	•	0	•	•	•	•	•	Z	•
LDY	•	•	•	N	•	•	•	•	•	Z	•
LDYB	•	•	•	0	•	•	•	•	•	Z	•
LSR	•	•	•	0	•	•	•	•	•	Z	C
LSR #n	•	•	•	0	•	•	•	•	•	Z	C
LSRD #n	•	•	•	0	•	•	•	•	•	Z	C

APPENDIX
Appendix 1. 7900 series machine instructions

Symbol	Function	Operation length (Bit)	Source	Destination DIR op	DIR n	DIR #	DIR, X op	DIR, X n	DIR, X #	ABS op	ABS n	ABS #	ABS, X op	ABS, X n	ABS, X #	Processor Status register 10 (IPL)	9 (IPL)	8 (IPL)	7 N	6 V	5 m	4 x	3 D	2 I	1 Z	0 C
MOVM **(Note 2)**	m = 0 M16(dest)←M16(source) m = 1 M8(dest)←M8(source)	16/8	IMM	86	5	3	31 47	7	4	96	4	4	31 57	6	5	•	•	•	•	•	•	•	•	•	•	•
			DIR	58	6	3				78	5	4														
			DIR, X							79	6	4														
			ABS	5C	6	4				7C	5	5														
			ABS, X	5D	7	4																				
MOVMB	M8(dest)←M8(source)	8	IMM	A9	5	3	31 3A	7	4	B9	4	4	31 3B	6	5	•	•	•	•	•	•	•	•	•	•	•
			DIR	48	6	3				68	5	4														
			DIR, X							69	6	4														
			ABS	4C	6	4				6C	5	5														
			ABS, X	4D	7	4																				
MOVR **(Notes 7 and 13)**	m = 0 M16(dest1) ←M16(source1) ⋮ ⋮ M16(dest n)←M16(source n) m = 1 M8(dest1) ←M8(source1) ⋮ ⋮ M8(dest n)←M8(source n) (n = 0 to 15)	16/8	IMM	61 10 + n	3 + 5n	2 + 2n				61 30 + n	3 + 4n	2 + 3n				•	•	•	•	•	•	•	•	•	•	•
			DIR	61 50 + n	3 + 6n	2 + 2n				61 70 + n	3 + 5n	2 + 3n														
			DIR, X							71 70 + n	3 + 6n	2 + 3n														
			ABS	61 90 + n	3 + 6n	2 + 3n				61 B0 + n	3 + 5n	2 + 4n														
			ABS, X	71 10 + n	3 + 6n	2 + 3n																				
MOVRB **(Note 7)**	M8(dest1) ←M8(source1) ⋮ ⋮ M8(dest n)←M8(source n) (n = to 15)	8	IMM	61 00 + n	3 + 5n	2 + 2n				61 20 + n	3 + 4n	2 + 3n				•	•	•	•	•	•	•	•	•	•	•
			DIR	61 40 + n	3 + 6n	2 + 2n				61 60 + n	3 + 5n	2 + 3n														
			DIR, X							71 60 + n	3 + 6n	2 + 3n														
			ABS	61 80 + n	3 + 6n	2 + 3n				61 A0 + n	3 + 5n	2 + 4n														
			ABS, X	71 00 + n	3 + 6n	2 + 3n																				

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes: IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y
			op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #
MPY **(Notes 2 and 14)**	(B, A)←A × M	16/8		31 C7 8 3		21 CA 9 3	21 CB 10 3		21 C0 11 3	21 C1 12 3	21 C8 12 3	21 C2 13 3	21 C9 14 3
MPYS **(Notes 2 and 14)**	(B, A)←A × M (Signed)	16/8		31 D7 8 3		21 DA 9 3	21 DB 10 3		21 D0 11 3	21 D1 12 3	21 D8 12 3	21 D2 13 3	21 D9 14 3
MVN **(Note 15)**	M(Y + k)←M(X + k) k = 0 to i – 1 (i: Number of transfer bytes specified by accumulator A)	16/8											
MVP **(Note 16)**	M(Y–k)←M(X–k) k = 0 to i–1 (i: Number of transfer bytes specified by accumulator A)	16/8											
NEG **(Note 1)**	Acc← –Acc	16/8			24 1 1 81 24 2 2								
NEGD	E← –E	32			31 80 4 2								
NOP	PC←PC + 1 When catty occurs in PC PG←PG + 1	–	74 1 1										
ORA **(Notes 1 and 2)**	Acc←Acc∨M	16/8		56 1 2 81 56 2 3		5A 3 2 81 5A 4 3	5B 4 2 81 5B 5 3		11 50 6 3 91 50 6 3	11 51 7 3 91 51 7 3	11 58 7 3 91 58 7 3	11 52 8 3 91 52 8 3	11 59 9 3 91 59 9 3
ORAB **(Note 1)**	Acc_L←Acc_L∨IMM8	8		63 1 2 81 63 2 3									
ORAM **(Note 3)**	M←M∨IMM	16/8				51 33 7 4							
ORAMB	M8←M8∨IMM8	8				51 32 7 4							
ORAMD	M32←M32∨IMM32	32				51 B3 10 7							
PEA	M(S)←IMM_H S←S – 1 M(S)←IMM_L S←S – 1	16											
PEI	M(S)←M((DPRn) + dd + 1) S←S + 1 M(S)←M((DPRn)+dd) S←S – 1 (n = 0 to 3)	16											

Addressing Modes: ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	Processor Status register: 10 9 8 7 6 5 4 3 2 1 0
op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	IPL N V m x D I Z C
21 CE 9 4	21 CF 10 4	21 C6 10 4	21 CC 10 5	21 CD 11 5								21 C3 10 3	21 C4 13 3			• • • N • • • • • Z 0
21 DE 9 4	21 DF 10 4	21 D6 10 4	21 DC 10 5	21 DD 11 5								21 D3 10 3	21 D4 13 3			• • • N • • • • • Z 0
														31 2B 5 + 5 i 4		• • • • • • • • • • •
														31 2A 5 + 5 i 4		• • • • • • • • • • •
																• • • N V • • • • Z C
																• • • N V • • • • Z C
																• • • • • • • • • • •
5E 3 3	5F 4 3	11 56 5 4	11 5C 5 5	11 5D 6 5								11 53 5 3	11 54 8 3			• • • N • • • • • Z •
81 5E 4 4	81 5F 5 4	91 56 5 4	91 5C 5 5	91 5D 6 5								91 53 5 3	91 54 8 3			
																• • • N • • • • • Z •
51 37 7 5																• • • N • • • • • Z •
51 36 7 5																• • • N • • • • • Z •
51 B7 10 8																• • • N • • • • • Z •
								31 4C 5 4								• • • • • • • • • • •
								31 4B 7 3								• • • • • • • • • • •

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes: STK op	STK n	STK #	Processor Status register: IPL (10)	IPL (9)	IPL (8)	N (7)	V (6)	m (5)	x (4)	D (3)	I (2)	Z (1)	C (0)
PER	EAR←PC + IMM16 M(S)←EAR$_H$ S←S – 1 M(S)←EAR$_L$ S←S – 1	16	31 4D	6	4	•	•	•	•	•	•	•	•	•	•	•
PHA	m = 0 M(S)←A$_H$ S←S – 1 M(S)←A$_L$ S←S – 1 m = 1 M(S)←A$_L$ S←S – 1	16/8	85	4	1	•	•	•	•	•	•	•	•	•	•	•
PHB	m = 0 M(S)←B$_H$ S←S – 1 M(S)←B$_L$ S←S – 1 m=1 M(S)←B$_L$ S←S – 1	16/8	81 85	5	2	•	•	•	•	•	•	•	•	•	•	•
PHD	M(S)←DPR0$_H$ S←S – 1 M(S)←DPR0$_L$ S←S – 1	16	83	4	1	•	•	•	•	•	•	•	•	•	•	•
PHD n **(Note 11)**	M(S)←DPRn$_H$ S←S – 1 M(S)←DPRn$_L$ S←S – 1 (n = 0 to 3) When multiple DPRs are specified, the above operations are repeated.	16	B8 01 0F B8 01 | 0F	12 11 +	2 2 i	•	•	•	•	•	•	•	•	•	•	•
PHG	M(S)←PG S←S – 1	8	31 60	4	2	•	•	•	•	•	•	•	•	•	•	•
PHLD n **(Note 11)**	M(S)←DPRn$_H$ S←S – 1 M(S)←DPRn$_L$ S←S – 1 DPRn←IMM16 (n = 0 to 3) When multiple DPRs are specified, the above operations are repeated.	16	B8 01 | 0F B8 01 | 0F	14 11 + 3 i	4 2 + 2 i	•	•	•	•	•	•	•	•	•	•	•
PHP	M(S)←PS$_H$ S←S – 1 M(S)←PS$_L$ S←S – 1	16	A5	4	1	•	•	•	•	•	•	•	•	•	•	•
PHT	M(S)←DT S←S – 1	8	31 40	4	2	•	•	•	•	•	•	•	•	•	•	•

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes: STK op	STK n	STK #	Processor Status register: 10 (IPL)	9 (IPL)	8 (IPL)	7 N	6 V	5 m	4 x	3 D	2 I	1 Z	0 C
PHX	x = 0 M(S)←X_H S←S – 1 M(S)←X_L S←S – 1 x = 1 M(S)←X_L S←S – 1	16/8	C5	4	1	•	•	•	•	•	•	•	•	•	•	•
PHY	x = 0 M(S)←Y_H S←S – 1 M(S)←Y_L S←S – 1 x = 1 M(S)←Y_L S←S – 1	16/8	E5	4	1	•	•	•	•	•	•	•	•	•	•	•
PLA	m = 0 S←S + 1 A_L←M(S) S←S + 1 A_H←M(S) m = 1 S←S + 1 A_L←M(S)	16/8	95	4	1	•	•	•	N	•	•	•	•	•	Z	•
PLB	m = 0 S←S + 1 B_L←M(S) S←S + 1 B_H←M(S) m = 1 S←S + 1 B_L←M(S)	16/8	81 95	5	2	•	•	•	N	•	•	•	•	•	Z	•
PLD	S←S + 1 $DPR0_L$←M(S) S←S + 1 $DPR0_H$←M(S)	16	93	5	1	•	•	•	•	•	•	•	•	•	•	•
PLD n **(Notes 11 and 12)**	S←S + 1 $DPRn_L$←M(S) S←S + 1 $DPRn_H$←M(S) (n = 0 to 3) When multiple DPRs are specified, the above operations are repeated.	16	77 ?0 77 ?0	11 8 + 3 i	2 2	•	•	•	•	•	•	•	•	•	•	•
PLP	S←S + 1 PS_L←M(S) S←S + 1 PS_H←M(S)	16	B5	5	1	Value restored from stack										
PLT	S←S + 1 DT←M(S)	8	31 50	6	2	•	•	•	N	•	•	•	•	•	Z	•

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	IMP (op n #)	IMM (op n #)	A (op n #)	DIR (op n #)	DIR, X (op n #)	DIR, Y (op n #)	(DIR) (op n #)	(DIR, X) (op n #)	(DIR), Y (op n #)	L(DIR) (op n #)	L(DIR), Y (op n #)
PLX	x = 0 S←S + 1 X_L←M(S) S←S + 1 X_H←M(S) x = 1 S←S + 1 X_L←M(S)	16/8											
PLY	x = 0 S←S + 1 Y_L←M(S) S←S + 1 Y_H←M(S) x = 1 S←S + 1 Y_L←M(S)	16/8											
PSH **(Note 17)**	M(S to S – i + 1)←A, B, X… S←S – i i: Number of bytes corresponding to register pushed on stack	16/8											
PUL **(Note 18)**	A, B, X…←M(S + 1 to S + i) S←S + i i: Number of bytes corresponding to register restored from stack	16/8											
RLA **(Note 3)**	Rotate to the left by n bits m = 0 (n = 0 to 65535) A: ← b_{15} … b_0 ← m = 1 (n = 0 to 255) A_L: ← b_7 … b_0 ←	16/8		31 07 / 5 + n / 3									
RMPA **(Note 19)**	m = 0 Repeat (B, A)←(B, A) + M(DT:X)✕ M(DT:Y) (Signed) X←X + 2 Y←Y + 2 i←i – 1 Until i = 0 m = 1 Repeat (B_L, A_L)←(B_L, A_L)+M(DT,X) M(DT,Y) (Signed) X←X + 1 Y←Y + 1 i←i – 1 Until i = 0 i: Numder of repetitions (0 to 255)	16/8											

Symbol	ABS (op n #)	ABS, X (op n #)	ABS, Y (op n #)	ABL (op n #)	ABL, X (op n #)	(ABS) (op n #)	L(ABS) (op n #)	(ABS, X) (op n #)	STK (op n #)	REL (op n #)	DIR, b, R (op n #)	ABS, b, R (op n #)	SR (op n #)	(SR), Y (op n #)	BLK (op n #)	MAA (op n #)	10 (IPL)	9 (IPL)	8 (IPL)	7 N	6 V	5 m	4 x	3 D	2 I	1 Z	0 C
PLX									D5 / 4 / 1								•	•	•	N	•	•	•	•	•	Z	•
PLY									F5 / 4 / 1								•	•	•	N	•	•	•	•	•	Z	•
PSH									A8 / 11 + 2i_1 + i_2 / 2								•	•	•	•	•	•	•	•	•	•	•
PUL									67 / 13 + 3 i / 2								When the contents of PS is restored, this becomes the value. In the other cases, nothing changes.										
RLA																	•	•	•	•	•	•	•	•	•	•	•
RMPA																31 5A 14 / 5 + imm / 3	•	•	•	N	V	•	•	•	•	Z	C

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes																																
			IMP			IMM			A			DIR			DIR, X			DIR, Y			(DIR)			(DIR, X)			(DIR), Y			L(DIR)			L(DIR), Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
ROL **(Note 1)**	Rotate to the left by 1 bit m = 0 Acc or M16 ← b_{15} … b_0 ← C ← m = 1 Acc_L or M8 ← b_7 … b_0 ← C ←	16/8							13 81 13	1 2	1 2	21 1A	7	3	21 1B	8	3																				
ROL #n **(Note 4)**	Rotate to the left by n bits (n = 0 to 15) m = 0 A ← b_{15} … b_0 ← C ← m = 1 A_L ← b_7 … b_0 ← C ←	16/8							C1 60 + imm	6 + imm	2																										
ROLD #n **(Note 4)**	Rotate to the left by n bits (n = 0 to 31) E ← b_{31} … b_0 ← C ←	32							D1 60 + imm	8 + imm	2																										
ROR **(Note 1)**	Rotate to the right by 1 bit m = 0 Acc or M16 → C → b_{15} … b_0 → m = 1 Acc_L or M8 → C → b_7 … b_0 →	16/8							53 81 53	1 2	1 2	21 3A	7	3	21 3B	8	3																				
ROR #n **(Note 4)**	Rotate to the right by n bits (n = 0 to 15) m = 0 A → C → b_{15} … b_0 → m = 1 A_L → C → b_7 … b_0 →	16/8							C1 20 + imm	6 + imm	2																										
RORD #n **(Note 4)**	Rotate to the right by n bits (n = 0 to 31) E → b_{31} … b_0 → C →	32							D1 20 + imm	8 + imm	2																										

Symbol	Addressing Modes																																															Processor Status register										
	ABS			ABS, X			ABS, Y			ABL			ABL, X			(ABS)			L(ABS)			(ABS, X)			STK			REL			DIR, b, R			ABS, b, R			SR			(SR), Y			BLK			MAA			10	9	8	7	6	5	4	3	2	1	0
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
ROL	21 1E	7	4	21 1F	8	4																																											•	•	•	N	•	•	•	•	•	Z	C
ROL #n																																																	•	•	•	N	•	•	•	•	•	Z	C
ROLD #n																																																	•	•	•	N	•	•	•	•	•	Z	C
ROR	21 3E	7	4	21 3F	8	4																																											•	•	•	N	•	•	•	•	•	Z	C
ROR #n																																																	•	•	•	N	•	•	•	•	•	Z	C
RORD #n																																																	•	•	•	N	•	•	•	•	•	Z	C

Addressing Modes (cells: op; n; #)

Symbol	Function	Operation length (Bit)	IMP	IMM	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y
RTI	S←S + 1 PSL←M(S) S←S + 1 PSH←M(S) S←S + 1 PCL←M(S) S←S + 1 PCH←M(S) S←S + 1 PG←M(S)	–	F1; 12; 1										
RTL	S←S + 1 PCL←M(S) S←S + 1 PCH←M(S) S←S + 1 PG←M(S)	–	94; 10; 1										
RTLD n **(Notes 11 and 12)**	S←S + 1 DPRnL←M(S) S←S + 1 DPRnH←M(S) S←S + 1 PCL←M(S) S←S + 1 PCH←M(S) S←S + 1 PG←M(S). (n = 0 to 3. Multiple DPRs can be specified.)	16											
RTS	S←S + 1 PCL←M(S) S←S + 1 PCH←M(S)	–	84; 7; 1										
RTSD n **(Notes 11 and 12)**	S←S + 1 DPRnL←M(S) S←S + 1 DPRnH←M(S) S←S + 1 PCL←M(S) S←S + 1 PCH←M(S), (n = 0 to 3. Multiple DPRs can be specified.)	16											
SBC **(Notes 1 and 2)**	Acc←Acc – M – $\overline{C}$	16/8		31 A7; 3; 3 B1 A7; 3; 3		21 AA; 5; 3 A1 AA; 7; 3	21 AB; 6; 3 A1 AB; 8; 3		21 A0; 7; 3 A1 A0; 9; 3	21 A1; 8; 3 A1 A1; 10; 3	21 A8; 8; 3 A1 A8; 10; 3	21 A2; 9; 3 A1 A2; 11; 3	21 A9; 10; 3 A1 A9; 12; 3
SBCB **(Note 1)**	AccL←AccL – IMM8 – $\overline{C}$	8		31 1B; 3; 3 B1 1B; 3; 3									
SBCD	E←E – M32 – $\overline{C}$	32		31 1D; 4; 6		21 BA; 7; 3	21 BB; 8; 3		21 B0; 9; 3	21 B1; 10; 3	21 B8; 10; 3	21 B2; 11; 3	21 B9; 12; 3
SEC	C←1	–	04; 1; 1										
SEI	I←1	–	05; 4; 1										

Addressing Modes (cells: op; n; #) and Processor Status register

Symbol	ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	Processor Status register (10 9 8: IPL; 7: N; 6: V; 5: m; 4: x; 3: D; 2: I; 1: Z; 0: C)
RTI																	Value restored from stack
RTL																	• • • • • • • • • • •
RTLD n									77 ?C; 15; 2 77 ?C; 12 + 3 i; 2								• • • • • • • • • • •
RTS																	• • • • • • • • • • •
RTSD n									77 ?8; 14; 2 77 ?8; 11 + 3 i; 2								• • • • • • • • • • •
SBC	21 AE; 5; 4 A1 AE; 7; 4	21 AF; 6; 4 A1 AF; 8; 4	21 A6; 6; 4 A1 A6; 8; 4	21 AC; 6; 5 A1 AC; 8; 5	21 AD; 7; 5 A1 AD; 9; 5								21 A3; 6; 3 A1 A3; 8; 3	21 A4; 9; 3 A1 A4; 11; 3			• • • N V • • • • Z C
SBCB																	• • • N V • • • • Z C
SBCD	21 BE; 7; 4	21 BF; 8; 4	21 B6; 8; 4	21 BC; 8; 5	21 BD; 9; 5								21 B3; 8; 3	21 B4; 11; 3			• • • N V • • • • Z C
SEC																	• • • • • • • • • • 1
SEI																	• • • • • • • • 1 • •

APPENDIX
Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	Addressing Modes																																
			IMP			IMM			A			DIR			DIR, X			DIR, Y			(DIR)			(DIR, X)			(DIR), Y			L(DIR)			L(DIR), Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
SEM	m←1	–	25	3	1																														
SEP	PS_L(bit n)←1 (n = 0 to 7. Multiple bits can be specified.)	–				99	3	2																											
STA **(Note 1)**	M←Acc	16/8										DA	4	2	DB	5	2				11 D0	7	3	11 D1	8	3	D8	7	2	11 D2	9	3	D9	9	2
												81 DA	5	3	81 DB	6	3				91 D0	7	3	91 D1	8	3	81 D8	8	3	91 D2	9	3	81 D9	10	3
STAB **(Note 1)**	M8←Acc_L	8										CA	4	2	CB	5	2				11 C0	7	3	11 C1	8	3	C8	7	2	11 C2	9	3	C9	9	2
												81 CA	5	3	81 CB	6	3				91 C0	7	3	91 C1	8	3	81 C8	8	3	91 C2	9	3	81 C9	10	3
STAD	M32←E	32										EA	6	2	EB	7	2				11 E0	9	3	11 E1	10	3	E8	9	2	11 E2	11	3	E9	11	2
STP	Oscillation stopped	–	31 30	–	2																														
STX	M←X	16/8										E2	4	2				41 E5	6 @	3 @	@														
STY	M 'Y	16/8										F2	4	2	41 FB	6	3																		
SUB **(Notes 1 and 2)**	Acc←Acc – M	16/8				36	1	2				3A	3	2	3B	4	2				11 30	6	3	11 31	7	3	11 38	7	3	11 32	8	3	11 39	9	3
						81 36	2	3				81 3A	4	3	81 3B	5	3				91 30	6	3	91 31	7	3	91 38	7	3	91 32	8	3	91 39	9	3
SUBB **(Note 1)**	Acc_L←Acc_L – IMM8	8				39	1	2																											
						81 39	2	3																											
SUBD	E←E – M32	32				3D	3	5				AA	6	2	AB	7	2				11 A0	9	3	11 A1	10	3	11 A8	10	3	11 A2	11	3	11 A9	12	3
SUBM **(Note 3)**	M←M – IMM	16/8										51 13	7	4																					
SUBMB	M8←M8 – IMM8	8										51 12	7	4																					
SUBMD	M32←M32 – IMM32	32										51 93	10	7																					

Symbol	Addressing Modes																																															
	ABS			ABS, X			ABS, Y			ABL			ABL, X			(ABS)			L(ABS)			(ABS, X)			STK			REL			DIR, b, R			ABS, b, R			SR			(SR), Y			BLK			MAA		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
SEM																																																
SEP																																																
STA	DE	4	3	DF	5	3	11 D6	6	4	DC	5	4	DD	6	4																						11 D3	6	3	11 D4	9	3						
	81 DE	5	4	81 DF	6	4	91 D6	6	4	81 DC	6	5	81 DD	7	5																						91 D3	6	3	91 D4	9	3						
STAB	CE	4	3	CF	5	3	11 C6	6	4	CC	5	4	CD	6	4																						11 C3	6	3	11 C4	9	3						
	81 CE	5	4	81 CF	6	4	91 C6	6	4	81 CC	6	5	81 CD	7	5																						91 C3	6	3	91 C4	9	3						
STAD	EE	6	3	EF	7	3	11 E6	8	4	EC	7	4	ED	8	4																						11 E3	8	3	11 E4	11	3						
STP																																																
STX	E7	4	3																																													
STY	F7	4	3																																													
SUB	3E	3	3	3F	4	3	11 36	5	4	11 3C	5	5	11 3D	6	5																						11 33	5	3	11 34	8	3						
	81 3E	4	4	81 3F	5	4	91 36	5	4	91 3C	5	5	91 3D	6	5																						91 33	5	3	91 34	8	3						
SUBB																																																
SUBD	AE	6	3	AF	7	3	11 A6	8	4	11 AC	8	5	11 AD	9	5																						11 A3	8	3	11 A4	11	3						
SUBM	51 17	7	5																																													
SUBMB	51 16	7	5																																													
SUBMD	51 97	10	8																																													

Symbol	Processor Status register										
	10	9	8	7	6	5	4	3	2	1	0
	IPL			N	V	m	x	D	I	Z	C
SEM	•	•	•	•	•	1	•	•	•	•	•
SEP	•	•	•	Specified flag becomes "1."							
STA	•	•	•	•	•	•	•	•	•	•	•
STAB	•	•	•	•	•	•	•	•	•	•	•
STAD	•	•	•	•	•	•	•	•	•	•	•
STP	•	•	•	•	•	•	•	•	•	•	•
STX	•	•	•	•	•	•	•	•	•	•	•
STY	•	•	•	•	•	•	•	•	•	•	•
SUB	•	•	•	N	V	•	•	•	•	Z	C
SUBB	•	•	•	N	V	•	•	•	•	Z	C
SUBD	•	•	•	N	V	•	•	•	•	Z	C
SUBM	•	•	•	N	V	•	•	•	•	Z	C
SUBMB	•	•	•	N	V	•	•	•	•	Z	C
SUBMD	•	•	•	N	V	•	•	•	•	Z	C

APPENDIX

Appendix 1. 7900 Series machine instructions

Symbol	Function	Operation length (Bit)	IMP op	IMP n	IMP #	IMM op	IMM n	IMM #	A	DIR	DIR, X	DIR, Y	(DIR)	(DIR, X)	(DIR), Y	L(DIR)	L(DIR), Y	ABS	ABS, X	ABS, Y	ABL	ABL, X	(ABS)	L(ABS)	(ABS, X)	STK	REL	DIR, b, R	ABS, b, R	SR	(SR), Y	BLK	MAA	IPL 10	IPL 9	IPL 8	N 7	V 6	m 5	x 4	D 3	I 2	Z 1	C 0
SUBS	S←S – IMM8	16				31 0B	2	3																										•	•	•	N	V	•	•	•	•	Z	C
SUBX **(Note 4)**	X←X – IMM (IMM = 0 to 31)	16/8				01 40 + imm	2	2																										•	•	•	N	V	•	•	•	•	Z	C
SUBY **(Note 4)**	Y←Y – IMM (IMM = 0 to 31)	16/8				01 60 + imm	2	2																										•	•	•	N	V	•	•	•	•	Z	C
TAD n **(Note 20)**	DPRn←A (n = 0 to 3)	16	31 n2	3	2																													•	•	•	•	•	•	•	•	•	•	•
TAS	S←A	16	31 82	2	2																													•	•	•	•	•	•	•	•	•	•	•
TAX	X←A	16/8	C4	1	1																													•	•	•	N	•	•	•	•	•	Z	•
TAY	Y←A	16/8	D4	1	1																													•	•	•	N	•	•	•	•	•	Z	•
TBD n **(Note 20)**	DPRn←B (n = 0 to 3)	16	B1 n2	3	2																													•	•	•	•	•	•	•	•	•	•	•
TBS	S←B	16	B1 82	2	2																													•	•	•	•	•	•	•	•	•	•	•
TBX	X←B	16/8	81 C4	2	2																													•	•	•	N	•	•	•	•	•	Z	•
TBY	Y←B	16/8	81 D4	2	2																													•	•	•	N	•	•	•	•	•	Z	•
TDA n **(Note 20)**	A←DPRn (n = 0 to 3)	16/8	31 40 + n2	2	2																													•	•	•	N	•	•	•	•	•	Z	•
TDB n **(Note 20)**	B←DPRn (n = 0 to 3)	16/8	B1 40 + n2	2	2																													•	•	•	N	•	•	•	•	•	Z	•
TDS	S←DPR0	16	31 73	2	2																													•	•	•	•	•	•	•	•	•	•	•

Symbol	Function	Operation length (Bit)	Addressing Modes																																
			IMP			IMM			A			DIR			DIR, X			DIR, Y			(DIR)			(DIR, X)			(DIR), Y			L(DIR)			L(DIR), Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
TSA	A←S	16/8	31 92	2	2																														
TSB	B←S	16/8	B1 92	2	2																														
TSD	DPR0←S	16	31 70	4	2																														
TSX	X←S	16/8	31 F2	2	2																														
TXA	A←X	16/8	A4	1	1																														
TXB	B←X	16/8	81 A4	2	2																														
TXS	S←X	16/8	31 E2	2	2																														
TXY	Y←X	16/8	31 C2	2	2																														
TYA	A←Y	16/8	B4	1	1																														
TYB	B←Y	16/8	81 B4	2	2																														
TYX	X←Y	16/8	31 D2	2	2																														
WIT	CPU clock stopped	–	31 10	–	2																														
XAB	A⇆B	16/8	55	2	1																														

Symbol	Addressing Modes																																															Processor Status register										
	ABS			ABS, X			ABS, Y			ABL			ABL, X			(ABS)			L(ABS)			(ABS, X)			STK			REL			DIR, b, R			ABS, b, R			SR			(SR), Y			BLK			MAA			10	9	8	7	6	5	4	3	2	1	0
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
TSA																																																	•	•	•	N	•	•	•	•	•	Z	•
TSB																																																	•	•	•	N	•	•	•	•	•	Z	•
TSD																																																	•	•	•	•	•	•	•	•	•	•	•
TSX																																																	•	•	•	N	•	•	•	•	•	Z	•
TXA																																																	•	•	•	N	•	•	•	•	•	Z	•
TXB																																																	•	•	•	N	•	•	•	•	•	Z	•
TXS																																																	•	•	•	•	•	•	•	•	•	•	•
TXY																																																	•	•	•	N	•	•	•	•	•	Z	•
TYA																																																	•	•	•	N	•	•	•	•	•	Z	•
TYB																																																	•	•	•	N	•	•	•	•	•	Z	•
TYX																																																	•	•	•	N	•	•	•	•	•	Z	•
WIT																																																	•	•	•	•	•	•	•	•	•	•	•
XAB																																																	•	•	•	N	•	•	•	•	•	Z	•

Notes for machine instructions table

This table lists the minimum number of instruction cycles for each instruction. The number of cycles of the addressing mode related with DPRn (n = 0 to 3) is applied when $DPRn_L$ = 0. When $DPRn_L \neq 0$, add 1 to the number of cycles.

The number of cycles also varies according to the number of bytes fetched into the instruction queue buffer, or according to whether the memory accessed is at an odd address or an even address. Furthermore, it also varies when the external area is accessed with BYTE = "H."

Note 1. The op code at the upper row is used for accumulator A, and the op code at the lower row is used for accumulator B.

Note 2. When handing 16-bit data with flag m = 0 in the IMM addressing mode, add 1 to the numder of bytes (#).

Note 3. When handing 16-bit data with flag m = 0, add 1 to the numder of bytes.

Note 4. Imm is the immediate value specified with an operand.

Note 5. The op code at the upper row is used for branching in the range of –128 to +127, and the op code at the lower row is used for branching in the range of –32768 to +32767.

Note 6. The BRK instruction is a reserved instruction for debugging tools; it cannot be used when an emulator is used.

Note 7. Any value from 0 through 15 is placed in an "n" in column "Addressing Modes."

Note 8. When handling 16-bit data with flag x = 0 in the IMM addressing mode, add 1 to the numder of bytes.

Note 9. The number of cycles is the case of the 16-bit ÷ 8-bit operation. In the case of the 32-bit ÷ 16-bit operation, add 8 to the number of cycles.

Note 10. When a zero division interrupt occurs, the number of cycles is 16 cycles. It is regardless of the data length.

Note 11. When placing a value in any of DPRs, the lower row is applied. When placing values to multiple DPRs, the lower row is applied. The letter "i" represents the number of DPRn specified: 1 to 4.

Note 12. A "?" indicates that the bit corressing to the specified DPRn becomes "1."

Note 13. When the source is in the addressing mode and flag m = 0, the number of bytes (#) is incremented by n (n = 0 to 15).

Note 14. The number of cycles of the case of the 8-bit × 8-bit operation. In the case of the 16-bit × 16-bit operation, add 4 to the number of cycles.

APPENDIX
Appendix 1. 7900 Series machine instructions

Note 15. The number of cycles is the case where the number of bytes to be transferred (#) is even. When the number of bytes to be transferred (#) is odd, the number is calculated as;

$$5 \times i + 10$$

Note 16. The number of cycles is the case where the number of bytes to be transferred (#) is even. When the number of bytes to be transferred (#) is odd, the number is calculated as;

$$5 \times i + 14$$

Note that it is 10 cycles in the case of 1-byte thanster.

Note 17. Add the number of cycles corresponding to the registers to be stored. i_1 is the number of registers to be stored among A, B, X, Y, DPR0, and PS. i_2 is the number of registers to be stored between DT and PG.

Note 18. Letter "i" indicates the number of registers to be restored.

Note 19. The number of cycles is applied when flag m = "1." When flag m="0," the number is calculated as;

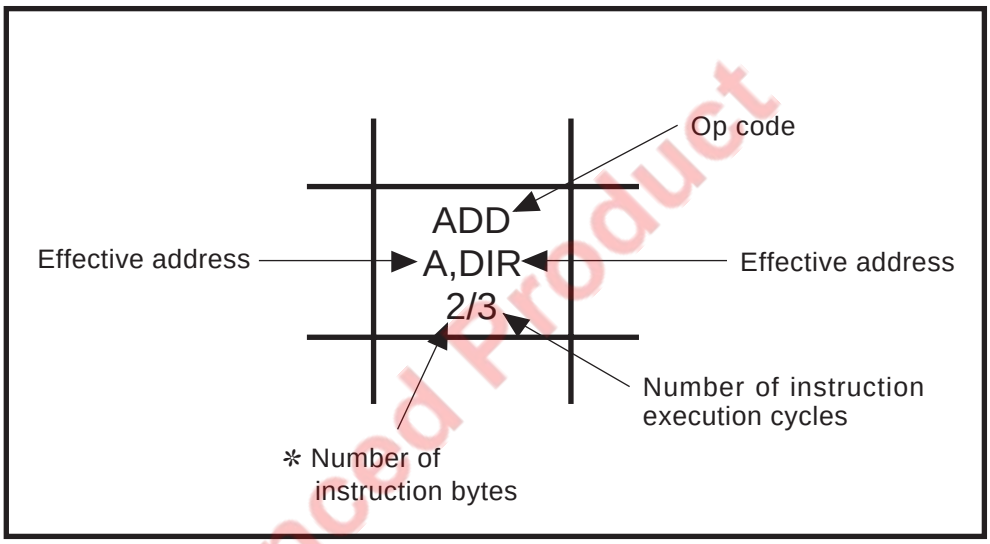
$$18 \times imm + 5$$

Note 20. Any value from 0 through 3 is placed in an "n" in column "Addressing Modes."

Appendix 2. Hexadecimal instruction code tables

[How to use these tables]

- First, see instruction code table 0-A.
- For an instruction of which op code consists of 2 bytes, the code corresponding to the 2nd byte is listed in another table. The 1st byte of the instruction listed in another table is indicated as “PAGE XX” in instruction code table 0-A.
- See the following:



✲ The inside of parentheses is applied when 16-bit data is handled with flag m = “0” or flag x= “0.” Unless otherwise noted, the instruction is unaffected by flags m and x.

Instruction code table 0-A

D7–D4 \ D3–D0 (Hexadecimal notation)		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	BRK (Note 1) IMP 2/15	PAGE10	LDX DIR 2/3	ASL A 1/1	SEC IMP 1/1	SEI IMP 1/4		LDX ABS 3/3	LDAB A,(DIR),Y 2/6	LDAB A,L(DIR),Y 2/8	LDAB A,DIR 2/3	LDAB A,DIR,X 2/4	LDAB A,ABL 4/4	LDAB A,ABL,X 4/5	LDAB A,ABS 3/3	LDAB A,ABS,X 3/4
0001	1	BPL REL 2/6	PAGE1-A	LDY DIR 2/3	ROL A 1/1	CLC IMP 1/1	CLI IMP 1/3	LDA A,IMM 2(3)/1	LDY ABS 3/3	LDA A,(DIR),Y 2/6	LDA A,L(DIR),Y 2/8	LDA A,DIR 2/3	LDA A,DIR,X 2/4	LDA A,ABL 4/4	LDA A,ABL,X 4/5	LDA A,ABS 3/3	LDA A,ABS,X 3/4
0010	2	BRA REL 2/5	PAGE2-A	CPX DIR 2/3	ANDB A,IMM 2/1	NEG A 1/1	SEM IMP 1/3	ADD A,IMM 2(3)/1	LDXB IMM 2/1	LDAB A,IMM 2/1	ADDB A,IMM 2/1	ADD A,DIR 2/3	ADD A,DIR,X 2/4	LDAD E,IMM 5/3	ADDD E,IMM 5/3	ADD A,ABS 3/3	ADD A,ABS,X 3/4
0011	3	BMI REL 2/6	PAGE3-A	CPY DIR 2/3	EORB A,IMM 2/1	EXTZ A 1/1	EXTS A 1/1	SUB A,IMM 2(3)/1	LDYB IMM 2/1	CMPB A,IMM 2/1	SUBB A,IMM 2/1	SUB A,DIR 2/3	SUB A,DIR,X 2/4	CMPD E,IMM 5/3	SUBD E,IMM 5/3	SUB A,ABS 3/3	SUB A,ABS,X 3/4
0100	4	BGTU REL 2/6	PAGE4	BBSB DIR,b,REL 4/8	LSR A 1/1	CLRB A 1/1	CLM IMP 1/3	CMP A,IMM 2(3)/1	BBSB ABS,b,REL 5/8	MOVMB DIR/DIR 3/6		CMP A,DIR 2/3	CMP A,DIR,X 2/4	MOVMB DIR/ABS 4/6	MOVMB DIR/ABS,X 4/7	CMP A,ABS 3/3	CMP A,ABS,X 3/4
0101	5	BVC REL 2/6	PAGE5	BBCB DIR,b,REL 4/8	ROR A 1/1	CLR A 1/1	XAB IMP 1/2	ORA A,IMM 2(3)/1	BBCB ABS,b,REL 5/8	MOVM DIR/DIR 3/6		ORA A,DIR 2/3	ORA A,DIR,X 2/4	MOVM DIR/ABS 4/6	MOVM DIR/ABS,X 4/7	ORA A,ABS 3/3	ORA A,ABS,X 3/4
0110	6	BLEU REL 2/6	PAGE6	CBEQB DIR/IMM,REL 4/8	ORAB A,IMM 2/1	ASR A 1/1	CLV IMP 1/1	AND A,IMM 2(3)/1	PUL STK 2/Note 2	MOVMB ABS/DIR 4/5	MOVMB ABS/DIR,X 4/6	AND A,DIR 2/3	AND A,DIR,X 2/4	MOVMB ABS/ABS 5/5		AND A,ABS 3/3	AND A,ABS,X 3/4
0111	7	BVS REL 2/6	PAGE7	CBNEB DIR/IMM,REL 4/8		NOP IMP 1/1		EOR A,IMM 2(3)/1	PLD n /RTLD n /RTSD n STK 2/Note 3	MOVM ABS/DIR 4/5	MOVM ABS/DIR,X 4/6	EOR A,DIR 2/3	EOR A,DIR,X 2/4	MOVM ABS/ABS 5/5		EOR A,ABS 3/3	EOR A,ABS,X 3/4
1000	8	BGT REL 2/6	PAGE0-B	INC DIR 2/6	PHD STK 1/4	RTS IMP 1/7	PHA STK 1/4	MOVM DIR/IMM 3(4)/5	INC ABS 3/6	LDAD E,(DIR),Y 2/9	LDAD E,L(DIR),Y 2/11	LDAD E,DIR 2/6	LDAD E,DIR,X 2/7	LDAD E,ABL 4/7	LDAD E,ABL,X 4/8	LDAD E,ABS 3/6	LDAD E,ABS,X 3/7
1001	9	BCC REL 2/6	PAGE1-B	DEC DIR 2/6	PLD STK 1/5	RTL IMP 1/10	PLA STK 1/4	MOVM ABS/IMM 4(5)/4	DEC ABS 3/6	CLP IMM 2/4	SEP IMM 2/3	ADDD E,DIR 2/6	ADDD E,DIR,X 2/7	JMP ABS 3/4	JSR ABS 3/6	ADDD E,ABS 3/6	ADDD E,ABS,X 3/7
1010	A	BLE REL 2/6	PAGE2-B	CBEQB A/IMM,REL 3/6	INC A 1/1	TXA IMP 1/1	PHP STK 1/4	CBEQ A/IMM,REL 3(4)/6	BRAL REL 3/5	PSH STK 2/Note 4	MOVMB DIR/IMM 3/5	SUBD E,DIR 2/6	SUBD E,DIR,X 2/7	JMPL ABL 4/5	JSRL ABL 4/7	SUBD E,ABS 3/6	SUBD E,ABS,X 3/7
1011	B	BCS REL 2/6	PAGE3-B	CBNEB A/IMM,REL 3/6	DEC A 1/1	TYA IMP 1/1	PLP STK 1/5	CBNE A/IMM,REL 3(4)/6		LDD n /PHD n /PHLD n STK/IMM /Note 5 and 6	MOVMB ABS/IMM 4/4	CMPD E,DIR 2/6	CMPD E,DIR,X 2/7	JMP (ABS,X) 3/7	JSR (ABS,X) 3/8	CMPD E,ABS 3/6	CMPD E,ABS,X 3/7
1100	C	BGE REL 2/6	PAGE8	CLRMB DIR 2/5	INX IMP 1/1	TAX IMP 1/1	PHX STK 1/4	LDX IMM 2(3)/1	CLRMB ABS 3/5	STAB A,(DIR),Y 2/7	STAB A,L(DIR),Y 2/9	STAB A,DIR 2/4	STAB A,DIR,X 2/5	STAB A,ABL 4/5	STAB A,ABL,X 4/6	STAB A,ABS 3/4	STAB A,ABS,X 3/5
1101	D	BNE REL 2/6	PAGE9	CLRM DIR 2/5	INY IMP 1/1	TAY IMP 1/1	PLX STK 1/4	LDY IMM 2(3)/1	CLRM ABS 3/5	STA A,(DIR),Y 2/7	STA A,L(DIR),Y 2/9	STA A,DIR 2/4	STA A,DIR,X 2/5	STA A,ABL 4/5	STA A,ABL,X 4/6	STA A,ABS 3/4	STA A,ABS,X 3/5
1110	E	BLT REL 2/6	ABS A 1/3	STX DIR 2/4	DEX IMP 1/1	CLRX IMP 1/1	PHY STK 1/4	CPX IMM 2(3)/1	STX ABS 3/4	STAD E,(DIR),Y 2/9	STAD E,L(DIR),Y 2/11	STAD E,DIR 2/6	STAD E,DIR,X 2/7	STAD E,ABL 4/7	STAD E,ABL,X 4/8	STAD E,ABS 3/6	STAD E,ABS,X 3/7
1111	F	BEQ REL 2/6	RTI IMP 1/12	STY DIR 2/4	DEY IMP 1/1	CLRY IMP 1/1	PLY STK 1/4	CPY IMM 2(3)/1	STY ABS 3/4	←			BSR REL 2/7				→

Instruction code table 1-A (PAGE 1-A)

D7–D4 \ D3–D0 (Hexadecimal notation)		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	LDAB A,(DIR) 3/6	LDAB A,(DIR,X) 3/7	LDAB A,L(DIR) 3/8	LDAB A,SR 3/5	LDAB A,(SR),Y 3/8		LDAB A,ABS,Y 4/5									
0001	1	LDA A,(DIR) 3/6	LDA A,(DIR,X) 3/7	LDA A,L(DIR) 3/8	LDA A,SR 3/5	LDA A,(SR),Y 3/8		LDA A,ABS,Y 4/5									
0010	2	ADD A,(DIR) 3/6	ADD A,(DIR,X) 3/7	ADD A,L(DIR) 3/8	ADD A,SR 3/5	ADD A,(SR),Y 3/8		ADD A,ABS,Y 4/5		ADD A,(DIR),Y 3/7	ADD A,L(DIR),Y 3/9			ADD A,ABL 5/5	ADD A,ABL,X 5/6		
0011	3	SUB A,(DIR) 3/6	SUB A,(DIR,X) 3/7	SUB A,L(DIR) 3/8	SUB A,SR 3/5	SUB A,(SR),Y 3/8		SUB A,ABS,Y 4/5		SUB A,(DIR),Y 3/7	SUB A,L(DIR),Y 3/9			SUB A,ABL 5/5	SUB A,ABL,X 5/6		
0100	4	CMP A,(DIR) 3/6	CMP A,(DIR,X) 3/7	CMP A,L(DIR) 3/8	CMP A,SR 3/5	CMP A,(SR),Y 3/8		CMP A,ABS,Y 4/5		CMP A,(DIR),Y 3/7	CMP A,L(DIR),Y 3/9			CMP A,ABL 5/5	CMP A,ABL,X 5/6		
0101	5	ORA A,(DIR) 3/6	ORA A,(DIR,X) 3/7	ORA A,L(DIR) 3/8	ORA A,SR 3/5	ORA A,(SR),Y 3/8		ORA A,ABS,Y 4/5		ORA A,(DIR),Y 3/7	ORA A,L(DIR),Y 3/9			ORA A,ABL 5/5	ORA A,ABL,X 5/6		
0110	6	AND A,(DIR) 3/6	AND A,(DIR,X) 3/7	AND A,L(DIR) 3/8	AND A,SR 3/5	AND A,(SR),Y 3/8		AND A,ABS,Y 4/5		AND A,(DIR),Y 3/7	AND A,L(DIR),Y 3/9			AND A,ABL 5/5	AND A,ABL,X 5/6		
0111	7	EOR A,(DIR) 3/6	EOR A,(DIR,X) 3/7	EOR A,L(DIR) 3/8	EOR A,SR 3/5	EOR A,(SR),Y 3/8		EOR A,ABS,Y 4/5		EOR A,(DIR),Y 3/7	EOR A,L(DIR),Y 3/9			EOR A,ABL 5/5	EOR A,ABL,X 5/6		
1000	8	LDAD E,(DIR) 3/9	LDAD E,(DIR,X) 3/10	LDAD E,L(DIR) 3/11	LDAD E,SR 3/8	LDAD E,(SR),Y 3/11		LDAD E,ABS,Y 4/8									
1001	9	ADDD E,(DIR) 3/9	ADDD E,(DIR,X) 3/10	ADDD E,L(DIR) 3/11	ADDD E,SR 3/8	ADDD E,(SR),Y 3/11		ADDD E,ABS,Y 4/8		ADDD E,(DIR),Y 3/10	ADDD E,L(DIR),Y 3/12			ADDD E,ABL 5/8	ADDD E,ABL,X 5/9		
1010	A	SUBD E,(DIR) 3/9	SUBD E,(DIR,X) 3/10	SUBD E,L(DIR) 3/11	SUBD E,SR 3/8	SUBD E,(SR),Y 3/11		SUBD E,ABS,Y 4/8		SUBD E,(DIR),Y 3/10	SUBD E,L(DIR),Y 3/12			SUBD E,ABL 5/8	SUBD E,ABL,X 5/9		
1011	B	CMPD E,(DIR) 3/9	CMPD E,(DIR,X) 3/10	CMPD E,L(DIR) 3/11	CMPD E,SR 3/8	CMPD E,(SR),Y 3/11		CMPD E,ABS,Y 4/8		CMPD E,(DIR),Y 3/10	CMPD E,L(DIR),Y 3/12			CMPD E,ABL 5/8	CMPD E,ABL,X 5/9		
1100	C	STAB A,(DIR) 3/7	STAB A,(DIR,X) 3/8	STAB A,L(DIR) 3/9	STAB A,SR 3/6	STAB A,(SR),Y 3/9		STAB A,ABS,Y 4/6									
1101	D	STA A,(DIR) 3/7	STA A,(DIR,X) 3/8	STA A,L(DIR) 3/9	STA A,SR 3/6	STA A,(SR),Y 3/9		STA A,ABS,Y 4/6									
1110	E	STAD E,(DIR) 3/9	STAD E,(DIR,X) 3/10	STAD E,L(DIR) 3/11	STAD E,SR 3/8	STAD E,(SR),Y 3/11		STAD E,ABS,Y 4/8									
1111	F																

APPENDIX

Appendix 2. Hexadecimal instruction code tables

Instruction code table 2-A (PAGE 2-A)

D7–D4 \ D3–D0 (Hexadecimal notation)		0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
0000	0											ASL DIR 3/7	ASL DIR,X 3/8			ASL ABS 4/7	ASL ABS,X 4/8
0001	1											ROL DIR 3/7	ROL DIR,X 3/8			ROL ABS 4/7	ROL ABS,X 4/8
0010	2											LSR DIR 3/7	LSR DIR,X 3/8			LSR ABS 4/7	LSR ABS,X 4/8
0011	3											ROR DIR 3/7	ROR DIR,X 3/8			ROR ABS 4/7	ROR ABS,X 4/8
0100	4											ASR DIR 3/7	ASR DIR,X 3/8			ASR ABS 4/7	ASR ABS,X 4/8
0101	5																
0110	6																
0111	7																
1000	8	ADC A,(DIR) 3/7	ADC A,(DIR,X) 3/8	ADC A,L(DIR) 3/9	ADC A,SR 3/6	ADC A,(SR),Y 3/9		ADC A,ABS,Y 4/6		ADC A,(DIR),Y 3/8	ADC A,L(DIR),Y 3/10	ADC A,DIR 3/5	ADC A,DIR,X 3/6	ADC A,ABL 5/6	ADC A,ABL,X 5/7	ADC A,ABS 4/5	ADC A,ABS,X 4/6
1001	9	ADCD E,(DIR) 3/9	ADCD E,(DIR,X) 3/10	ADCD E,L(DIR) 3/11	ADCD E,SR 3/8	ADCD E,(SR),Y 3/11		ADCD E,ABS,Y 4/8		ADCD E,(DIR),Y 3/10	ADCD E,L(DIR),Y 3/12	ADCD E,DIR 3/7	ADCD E,DIR,X 3/8	ADCD E,ABL 5/8	ADCD E,ABL,X 5/9	ADCD E,ABS 4/7	ADCD E,ABS,X 4/8
1010	A	SBC A,(DIR) 3/7	SBC A,(DIR,X) 3/8	SBC A,L(DIR) 3/9	SBC A,SR 3/6	SBC A,(SR),Y 3/9		SBC A,ABS,Y 4/6		SBC A,(DIR),Y 3/8	SBC A,L(DIR),Y 3/10	SBC A,DIR 3/5	SBC A,DIR,X 3/6	SBC A,ABL 5/6	SBC A,ABL,X 5/7	SBC A,ABS 4/5	SBC A,ABS,X 4/6
1011	B	SBCD E,(DIR) 3/9	SBCD E,(DIR,X) 3/10	SBCD E,L(DIR) 3/11	SBCD E,SR 3/8	SBCD E,(SR),Y 3/11		SBCD E,ABS,Y 4/8		SBCD E,(DIR),Y 3/10	SBCD E,L(DIR),Y 3/12	SBCD E,DIR 3/7	SBCD E,DIR,X 3/8	SBCD E,ABL 5/8	SBCD E,ABL,X 5/9	SBCD E,ABS 4/7	SBCD E,ABS,X 4/8
1100	C	MPY (DIR) 3/11/Note 7	MPY (DIR,X) 3/12/Note 7	MPY L(DIR) 3/13/Note 7	MPY SR 3/10/Note 7	MPY (SR),Y 3/13/Note 7		MPY ABS,Y 4/10/Note 7		MPY (DIR),Y 3/12/Note 7	MPY L(DIR),Y 3/14/Note 7	MPY DIR 3/9/Note 7	MPY DIR,X 3/10/Note 7	MPY ABL 5/10/Note 7	MPY ABL,X 5/11/Note 7	MPY ABS 4/9/Note 7	MPY ABS,X 4/10/Note 7
1101	D	MPYS (DIR) 3/11/Note 7	MPYS (DIR,X) 3/12/Note 7	MPYS L(DIR) 3/13/Note 7	MPYS SR 3/10/Note 7	MPYS (SR),Y 3/13/Note 7		MPYS ABS,Y 4/10/Note 7		MPYS (DIR),Y 3/12/Note 7	MPYS L(DIR),Y 3/14/Note 7	MPYS DIR 3/9/Note 7	MPYS DIR,X 3/10/Note 7	MPYS ABL 5/10/Note 7	MPYS ABL,X 5/11/Note 7	MPYS ABS 4/9/Note 7	MPYS ABS,X 4/10/Note 7
1110	E	DIV (DIR) 3/18/Note 8,9	DIV (DIR,X) 3/19/Note 8,9	DIV L(DIR) 3/20/Note 8,9	DIV SR 3/17/Note 8,9	DIV (SR),Y 3/20/Note 8,9		DIV ABS,Y 4/17/Note 8,9		DIV (DIR),Y 3/19/Note 8,9	DIV L(DIR),Y 3/21/Note 8,9	DIV DIR 3/16/Note 8,9	DIV DIR,X 3/17/Note 8,9	DIV ABL 5/17/Note 8,9	DIV ABL,X 5/18/Note 8,9	DIV ABS 4/16/Note 8,9	DIV ABS,X 4/17/Note 8,9
1111	F	DIVS (DIR) 3/25/Note 8,9	DIVS (DIR,X) 3/26/Note 8,9	DIVS L(DIR) 3/27/Note 8,9	DIVS SR 3/24/Note 8,9	DIVS (SR),Y 3/27/Note 8,9		DIVS ABS,Y 4/24/Note 8,9		DIVS (DIR),Y 3/26/Note 8,9	DIVS L(DIR),Y 3/28/Note 8,9	DIVS DIR 3/23/Note 8,9	DIVS DIR,X 3/24/Note 8,9	DIVS ABL 5/24/Note 8,9	DIVS ABL,X 5/25/Note 8,9	DIVS ABS 4/23/Note 8,9	DIVS ABS,X 4/24/Note 8,9

Instruction code table 3-A (PAGE 3-A)

D7–D4 \ D3–D0 (Hexadecimal notation)		0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
0000	0			TAD,0 IMP 2/3					RLA A 3(4)/n+5 /Note 10			ADDS IMM 3/2	SUBS IMM 3/2				
0001	1	WIT IMP 2/-		TAD,1 IMP 2/3								ADCB A,IMM 3/3	SBCB A,IMM 3/3	ADCD E,IMM 6/4	SBCD E,IMM 6/4		
0010	2			TAD,2 IMP 2/3								MVP BLK 4/5i+9/Note 11	MVN BLK 4/5i+5/Note 12				
0011	3	STP IMP 2/-		TAD,3 IMP 2/3								MOVMB DIR,X/IMM 4/7	MOVMB ABS,X/IMM 5/6				
0100	4	PHT STK 2/4		TDA,0 IMP 2/2					MOVM DIR,X/IMM 4(5)/7			LDT IMM 3/4	PEI STK 3/7	PEA STK 4/5	PER STK 4/6		
0101	5	PLT STK 2/6		TDA,1 IMP 2/2					MOVM ABS,X/IMM 5(6)/6			RMPA Multiplied accumulation 3/14imm+5 /Note 13		JMP (ABS) 4/7	JMPL (ABS) 4/9		
0110	6	PHG STK 2/4		TDA,2 IMP 2/2													
0111	7	TSD IMP 2/4		TDA,3 IMP 2/2	TDS IMP 2/2												
1000	8	NEGD E 2/4		TAS IMP 2/2					ADC A,IMM 3(4)/3								
1001	9	ABSD E 2/5		TSA IMP 2/2													
1010	A	EXTZD E 2/3							SBC A,IMM 3(4)/3								
1011	B	EXTSD E 2/5															
1100	C			TXY IMP 2/2					MPY IMM 3(4)/8/Note 7								
1101	D			TYX IMP 2/2					MPYS IMM 3(4)/8/Note 7								
1110	E			TXS IMP 2/2					DIV IMM 3(4)/15/Note 8,9								
1111	F			TSX IMP 2/2					DIVS IMM 3(4)/22/Note 8,9								

Instruction code table 4 (PAGE 4)

D7–D4 \ D3–D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0						LDX DIR,Y 3/5	LDX ABS,Y 4/5									
0001	1												LDY DIR,X 3/5				LDY ABS,X 4/5
0010	2															CPX ABS 4/4	
0011	3															CPY ABS 4/4	
0100	4											BBS DIR,b,REL 5(6)/9				BBS ABS,b,REL 6(7)/9	
0101	5											BBC DIR,b,REL 5(6)/9				BBC ABS,b,REL 6(7)/9	
0110	6											CBEQ DIR/IMM,REL 5(6)/9					
0111	7											CBNE DIR/IMM,REL 5(6)/9					
1000	8												INC DIR,X 3/8				INC ABS,X 4/8
1001	9												DEC DIR,X 3/8				DEC ABS,X 4/8
1010	A																
1011	B																
1100	C																
1101	D																
1110	E						STX DIR,Y 3/6										
1111	F												STY DIR,X 3/6				

Instruction code table 5 (PAGE 5)

D7–D4 \ D3–D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0			ADDMB DIR/IMM 4/7	ADDM DIR/IMM 4(5)/7			ADDMB ABS/IMM 5/7	ADDM ABS/IMM 5(6)/7								
0001	1			SUBMB DIR/IMM 4/7	SUBM DIR/IMM 4(5)/7			SUBMB ABS/IMM 5/7	SUBM ABS/IMM 5(6)/7								
0010	2			CMPMB DIR/IMM 4/5	CMPM DIR/IMM 4(5)/5			CMPMB ABS/IMM 5/5	CMPM ABS/IMM 5(6)/5								
0011	3			ORAMB DIR/IMM 4/7	ORAM DIR/IMM 4(5)/7			ORAMB ABS/IMM 5/7	ORAM ABS/IMM 5(6)/7								
0100	4																
0101	5																
0110	6			ANDMB DIR/IMM 4/7	ANDM DIR/IMM 4(5)/7			ANDMB ABS/IMM 5/7	ANDM ABS/IMM 5(6)/7								
0111	7			EORMB DIR/IMM 4/7	EORM DIR/IMM 4(5)/7			EORMB ABS/IMM 5/7	EORM ABS/IMM 5(6)/7								
1000	8				ADDMD DIR/IMM 7/10				ADDMD ABS/IMM 8/10								
1001	9				SUBMD DIR/IMM 7/10				SUBMD ABS/IMM 8/10								
1010	A				CMPMD DIR/IMM 7/7				CMPMD ABS/IMM 8/7								
1011	B				ORAMD DIR/IMM 7/10				ORAMD ABS/IMM 8/10								
1100	C																
1101	D																
1110	E				ANDMD DIR/IMM 7/10				ANDMD ABS/IMM 8/10								
1111	F				EORMD DIR/IMM 7/10				EORMD ABS/IMM 8/10								

APPENDIX

Appendix 2. Hexadecimal instruction code tables

Instruction code table 6 (PAGE 6)

D7–D4 \ D3–D0 (Hexadecimal notation)		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	MOVRB DIR/IMM 2n+2/5n+3 /Note 14															→
0001	1	MOVR DIR/IMM 2n(3n)+2/5n+3 /Note 14															→
0010	2	MOVRB ABS/IMM 3n+2/4n+3/ Note 14															→
0011	3	MOVR ABS/IMM 3n(4n)+2/4n+3 /Note 14															→
0100	4	MOVRB DIR/DIR 2n+2/6n+3 /Note 14															→
0101	5	MOVR DIR/DIR 2n+2/6n+3 /Note 14															→
0110	6	MOVRB ABS/DIR 3n+2/5n+3 /Note 14															→
0111	7	MOVR ABS/DIR 3n+2/5n+3 /Note 14															→
1000	8	MOVRB DIR/ABS 3n+2/6n+3 /Note 14															→
1001	9	MOVR DIR/ABS 3n+2/6n+3 /Note 14															→
1010	A	MOVRB ABS/ABS 4n+2/5n+3 /Note 14															→
1011	B	MOVR ABS/ABS 4n+2/5n+3 /Note 14															→
1100	C																
1101	D																
1110	E																
1111	F																

Instruction code table 7 (PAGE 7)

D7–D4 \ D3–D0 (Hexadecimal notation)		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	MOVRB DIR/ABS,X 3n+2/6n+3 /Note 14															→
0001	1	MOVR DIR/ABS,X 3n+2/6n+3 /Note 14															→
0010	2																
0011	3																
0100	4																
0101	5																
0110	6	MOVRB ABS/DIR,X 3n+2/6n+3 /Note 14															→
0111	7	MOVR ABS/DIR,X 3n+2/6n+3 /Note 14															→
1000	8								BSS DIR,b,REL 4/11								
1001	9																
1010	A								BSC DIR,b,REL 4/11								
1011	B																
1100	C								BSS ABS,b,REL 5/10								
1101	D																
1110	E								BSC ABS,b,REL 5/10								
1111	F																

Instruction code table 8 (PAGE 8)

D7–D4 \ D3–D0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal notation	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000								LSR,#n A 2/imm+6 /Note 15								
0001																
0010								ROR,#n A 2/imm+6 /Note 15								
0011																
0100								ASL,#n A 2/imm+6 /Note 15								
0101																
0110								ROL,#n A 2/imm+6 /Note 15								
0111																
1000								ASR,#n A 2/imm+6 /Note 15								
1001																
1010								DEBNE DIR/IMM,REL 4/12								
1011																
1100																
1101																
1110																
1111																

Instruction code table 9 (PAGE 9)

D7–D4 \ D3–D0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal notation	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000								LSRD,#n E 2/imm+8 /Note 16								
0001																
0010								RORD,#n E 2/imm+8 /Note 16								
0011																
0100								ASLD,#n E 2/imm+8 /Note 16								
0101																
0110								ROLD,#n E 2/imm+8 /Note 16								
0111																
1000								ASRD,#n E 2/imm+8 /Note 16								
1001																
1010																
1011																
1100																
1101																
1110								DEBNE ABS/IMM,REL 5/11								
1111																

APPENDIX

Appendix 2. Hexadecimal instruction code tables

Instruction code table 10 (PAGE 10)

D7–D4 \ D3–D0 (Hexadecimal notation)	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (A)	1011 (B)	1100 (C)	1101 (D)	1110 (E)	1111 (F)
0000	0															
0001	1															
0010	2															
0011	3															
0100	4															
0101	5															
0110	6															
0111	7															
1000	8															
1001	9															
1010	A															
1011	B															
1100	C															
1101	D															
1110	E															
1111	F															

ADDX IMM 2/2

ADDY IMM 2/2

SUBX IMM 2/2

SUBY IMM 2/2

BSS A,b,REL 3/7

BSC A,b,REL 3/7

DXBNE IMM,REL 3/7

DYBNE IMM,REL 3/7

Instruction code table 0-B (PAGE 0-B)

D7–D4 \ D3–D0 (Hexadecimal notation)	0000 (0)	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (A)	1011 (B)	1100 (C)	1101 (D)	1110 (E)	1111 (F)
0000	0				ASL B 2/2				LDAB B,(DIR),Y 3/7	LDAB B,L(DIR),Y 3/9	LDAB B,DIR 3/4	LDAB B,DIR,X 3/5	LDAB B,ABL 5/5	LDAB B,ABL,X 5/6	LDAB B,ABS 4/4	LDAB B,ABS,X 4/5
0001	1				ROL B 2/2		LDA B,IMM 3(4)/2		LDA B,(DIR),Y 3/7	LDA B,L(DIR),Y 3/9	LDA B,DIR 3/4	LDA B,DIR,X 3/5	LDA B,ABL 5/5	LDA B,ABL,X 5/6	LDA B,ABS 4/4	LDA B,ABS,X 4/5
0010	2				ANDB B,IMM 3/2	NEG B 2/2	ADD B,IMM 3(4)/2		LDAB B,IMM 3/2	ADDB B,IMM 3/2	ADD B,DIR 3/4	ADD B,DIR,X 3/5			ADD B,ABS 4/4	ADD B,ABS,X 4/5
0011	3				EORB B,IMM 3/2	EXTZ B 2/2	EXTS B 2/2	SUB B,IMM 3(4)/2	CMPB B,IMM 3/2	SUBB B,IMM 3/2	SUB B,DIR 3/4	SUB B,DIR,X 3/5			SUB B,ABS 4/4	SUB B,ABS,X 4/5
0100	4				LSR B 2/2	CLRB B 2/2	CMP B,IMM 3(4)/2				CMP B,DIR 3/4	CMP B,DIR,X 3/5			CMP B,ABS 4/4	CMP B,ABS,X 4/5
0101	5				ROR B 2/2	CLR B 2/2	ORA B,IMM 3(4)/2				ORA B,DIR 3/4	ORA B,DIR,X 3/5			ORA B,ABS 4/4	ORA B,ABS,X 4/5
0110	6				ORAB B,IMM 3/2	ASR B 2/2	AND B,IMM 3(4)/2				AND B,DIR 3/4	AND B,DIR,X 3/5			AND B,ABS 4/4	AND B,ABS,X 4/5
0111	7						EOR B,IMM 3(4)/2				EOR B,DIR 3/4	EOR B,DIR,X 3/5			EOR B,ABS 4/4	EOR B,ABS,X 4/5
1000	8						PHB STK 2/5									
1001	9						PLB STK 2/5									
1010	A			CBEQB B/IMM,REL 4/7	INC B 2/2	TXB IMP 2/2	CBEQ B/IMM,REL 4(5)/7									
1011	B			CBNEB B/IMM,REL 4/7	DEC B 2/2	TYB IMP 2/2	CBNE B/IMM,REL 4(5)/7									
1100	C					TBX IMP 2/2			STAB B,(DIR),Y 3/8	STAB B,L(DIR),Y 3/10	STAB B,DIR 3/5	STAB B,DIR,X 3/6	STAB B,ABL 5/6	STAB B,ABL,X 5/7	STAB B,ABS 4/5	STAB B,ABS,X 4/6
1101	D					TBY IMP 2/2			STA B,(DIR),Y 3/8	STA B,L(DIR),Y 3/10	STA B,DIR 3/5	STA B,DIR,X 3/6	STA B,ABL 5/6	STA B,ABL,X 5/7	STA B,ABS 4/5	STA B,ABS,X 4/6
1110	E		ABS B 2/4													
1111	F															

Instruction code table 1-B (PAGE 1-B)

D7–D4 \ D3–D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	LDAB B,(DIR) 3/6	LDAB B,(DIR,X) 3/7	LDAB B,L(DIR) 3/8	LDAB B,SR 3/5	LDAB B,(SR),Y 3/8		LDAB B,ABS,Y 4/5									
0001	1	LDA B,(DIR) 3/6	LDA B,(DIR,X) 3/7	LDA B,L(DIR) 3/8	LDA B,SR 3/5	LDA B,(SR),Y 3/8		LDA B,ABS,Y 4/5									
0010	2	ADD B,(DIR) 3/6	ADD B,(DIR,X) 3/7	ADD B,L(DIR) 3/8	ADD B,SR 3/5	ADD B,(SR),Y 3/8		ADD B,ABS,Y 4/5		ADD B,(DIR),Y 3/7	ADD B,L(DIR),Y 3/9			ADD B,ABL 5/5	ADD B,ABL,X 5/6		
0011	3	SUB B,(DIR) 3/6	SUB B,(DIR,X) 3/7	SUB B,L(DIR) 3/8	SUB B,SR 3/5	SUB B,(SR),Y 3/8		SUB B,ABS,Y 4/5		SUB B,(DIR),Y 3/7	SUB B,L(DIR),Y 3/9			SUB B,ABL 5/5	SUB B,ABL,X 5/6		
0100	4	CMP B,(DIR) 3/6	CMP B,(DIR,X) 3/7	CMP B,L(DIR) 3/8	CMP B,SR 3/5	CMP B,(SR),Y 3/8		CMP B,ABS,Y 4/5		CMP B,(DIR),Y 3/7	CMP B,L(DIR),Y 3/9			CMP B,ABL 5/5	CMP B,ABL,X 5/6		
0101	5	ORA B,(DIR) 3/6	ORA B,(DIR,X) 3/7	ORA B,L(DIR) 3/8	ORA B,SR 3/5	ORA B,(SR),Y 3/8		ORA B,ABS,Y 4/5		ORA B,(DIR),Y 3/7	ORA B,L(DIR),Y 3/9			ORA B,ABL 5/5	ORA B,ABL,X 5/6		
0110	6	AND B,(DIR) 3/6	AND B,(DIR,X) 3/7	AND B,L(DIR) 3/8	AND B,SR 3/5	AND B,(SR),Y 3/8		AND B,ABS,Y 4/5		AND B,(DIR),Y 3/7	AND B,L(DIR),Y 3/9			AND B,ABL 5/5	AND B,ABL,X 5/6		
0111	7	EOR B,(DIR) 3/6	EOR B,(DIR,X) 3/7	EOR B,L(DIR) 3/8	EOR B,SR 3/5	EOR B,(SR),Y 3/8		EOR B,ABS,Y 4/5		EOR B,(DIR),Y 3/7	EOR B,L(DIR),Y 3/9			EOR B,ABL 5/5	EOR B,ABL,X 5/6		
1000	8																
1001	9																
1010	A																
1011	B																
1100	C	STAB B,(DIR) 3/7	STAB B,(DIR,X) 3/8	STAB B,L(DIR) 3/9	STAB B,SR 3/6	STAB B,(SR),Y 3/9		STAB B,ABS,Y 4/6									
1101	D	STA B,(DIR) 3/7	STA B,(DIR,X) 3/8	STA B,L(DIR) 3/9	STA B,SR 3/6	STA B,(SR),Y 3/9		STA B,ABS,Y 4/6									
1110	E																
1111	F																

Instruction code table 2-B (PAGE 2-B)

D7–D4 \ D3–D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0																
0001	1																
0010	2																
0011	3																
0100	4																
0101	5																
0110	6																
0111	7																
1000	8	ADC B,(DIR) 3/9	ADC B,(DIR,X) 3/10	ADC B,L(DIR) 3/11	ADC B,SR 3/8	ADC B,(SR),Y 3/11		ADC B,ABS,Y 4/8		ADC B,(DIR),Y 3/10	ADC B,L(DIR),Y 3/12	ADC B,DIR 3/7	ADC B,DIR,X 3/8	ADC B,ABL 5/8	ADC B,ABL,X 5/9	ADC B,ABS 4/7	ADC B,ABS,X 4/8
1001	9																
1010	A	SBC B,(DIR) 3/9	SBC B,(DIR,X) 3/10	SBC B,L(DIR) 3/11	SBC B,SR 3/8	SBC B,(SR),Y 3/11		SBC B,ABS,Y 4/8		SBC B,(DIR),Y 3/10	SBC B,L(DIR),Y 3/12	SBC B,DIR 3/7	SBC B,DIR,X 3/8	SBC B,ABL 5/8	SBC B,ABL,X 5/9	SBC B,ABS 4/7	SBC B,ABS,X 4/8
1011	B																
1100	C																
1101	D																
1110	E																
1111	F																

APPENDIX

Appendix 2. Hexadecimal instruction code tables

Instruction code table 3-B (PAGE 3-B)

D7–D4 \ D3–D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0			TBD,0 IMP 2/3													
0001	1			TBD,1 IMP 2/3								ADCB B,IMM 3/3	SBCB B,IMM 3/3				
0010	2			TBD,2 IMP 2/3													
0011	3			TBD,3 IMP 2/3													
0100	4			TDB,0 IMP 2/2													
0101	5			TDB,1 IMP 2/2													
0110	6			TDB,2 IMP 2/2													
0111	7			TDB,3 IMP 2/2													
1000	8			TBS IMP 2/2					ADC B,IMM 3(4)/3								
1001	9			TSB IMP 2/2													
1010	A								SBC B,IMM 3(4)/3								
1011	B																
1100	C																
1101	D																
1110	E																
1111	F																

Notes for machine instructions table

This table lists the minimum number of instruction cycles for each instruction. The number of cycles of the addressing mode related with DPRn (n = 0 to 3) is applied when $DPRn_L = 0$. When $DPRn_L \neq 0$, add 1 to the number of cycles.

The number of cycles also varies according to the number of bytes fetched into the instruction queue buffer, or according to whether the memory accessed is at an odd address or an even address. Furthermore, it also varies when the external area is accessed with BYTE="H."

Note 1. The BRK instruction is a reserved instruction for debugging tools; it cannot be used when an emulator is used.

Note 2. 3i + 13 i is the number of registers to be restored.

Note 3. PLDn : 11, PLD (n_1, ..., n_i) : 3i + 8 (n_1, ..., n_i) : 0 to 3 (numbers representing DPRn)
RTLDn : 15, RTLD (n_1, ..., n_i) : 3i + 12 i is the number of DPRs specified (1 to 4).
RTSDn : 14, RTSD (n_1, ..., n_i) : 3i + 11

Note 4. $2i_1 + i_2 + 11$ Add the number of cycles corresponding to the registers to be stored. i_1 is the number of registers to be stored among A, B, X, Y, DPR0, and PS. i_2 is the number of registers to be stored between DT and PG.

Note 5. LDDn : 4, LDD (n_1, ..., n_i) : 2i + 2 (n_1, ..., n_i) : 0 to 3 (numbers representing DPRn)
PHDn : 2, PHD (n_1, ..., n_i) : 2 i is the number of DPRs specified (1 to 4).
PHLDn : 4, PHLD (n_1, ..., n_i) : 2i + 2

Note 6. LDDn : 13, LDD (n_1, ..., n_i) : 2i + 11 (n_1, ..., n_i) : 0 to 3 (numbers representing DPRn)
PHDn : 12, PHD (n_1, ..., n_i) : i + 11 i is the number of DPRs specified (1 to 4).
PHLDn : 14, PHLD (n_1, ..., n_i) : 3i + 11

Note 7. The number of cycles is the case of the 8-bit × 8-bit operation. Add 4 to the number of cycles in the case of the 16-bit × 16-bit operation.

Note 8. The number of cycles is the case of the 16-bit ÷ 8-bit operation. Add 8 to the number of cycles in the case of the 32-bit ÷ 16-bit operation.

Note 9. When a zero division interrupt occurs, the number of cycles is 16 cycles. It is regardless of the data length.

Note 10. n is the number of rotation specified by imm.
m = 0 : n = 0 to 65535
m = 1 : n = 0 to 255

Note 11. The number of cycles is the case where the number of bytes to be transferred (#) is even. When the number of bytes to be transferred (#) is odd, the number is calculated as;

5 × i + 14

Note that it is 10 cycles in the case of 1-byte transfer.

Note 12. The number of cycles is the case where the number of bytes to be transferred (#) is even. When the number of bytes to be transferred (#) is odd, the number is calculated as;

5 × i + 10

Note 13. The number of cycles is the case where flag m="1." When flag m="0," the number is calculated as;

18 × imm + 5 (imm = number of repeat times, 0 to 255)

Note 14. n = 0 to 15

Note 15. imm = 0 to 15

Note 16. imm = 0 to 31

MITSUBISHI SEMICONDUCTORS
Software Manual
7900 Series

Jul., First Edition 1998

Editioned by
Committee of editing of Mitsubishi Semiconductor Software Manual

Published by
Mitsubishi Electric Corp., Semiconductor Marketing Division

7900 Series
Software Manual

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan