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.

User's Manual

RENESAS

Phase-out/Discontinued

17K Series

4-bit Singlechip Microcontrollers

Instructions

Document No. U12986EJ2V0UM00 (2nd edition)
(Previous No. IEU-1312)
Date Published September 1997

Phase-out/Discontinued

[MEMO]

The information in this document is subject to change without notice.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.

While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.

NEC devices are classified into the following three quality grades:

"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

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

Special: 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: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books.

If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

Anti-radioactive design is not implemented in this product.

Phase-out/Discontinued

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

Major Revisions in This Version

Section	Description
Whole manual	All program examples amended and added
2-1	Addition of Chapter 2 "Program Memory (ROM)"
3-1	Addition of Chapter 3 "Program Counter (PC)"
5-109	Addition of 5.40 "SYSCAL entry"

PREFACE

Intended readership : This manual is intended for engineers who require an understanding of 17K series functions prior to designing an application system using this software.

Purpose : The purpose of this manual is to help engineers to understand the various instruction functions with which the 17K series is provided.

Organization: This manual broadly consists of the following contents.

. General description
. Program memory (ROM)
. Program counter (PC)
. Data memory address specification method
. Instruction set

CONTENTS

CHAPTER 1. GENERAL DESCRIPTION

1.1 OUTLINE

17K series instructions all have a single 16-bit word structure, allowing efficient programming. The instruction set comprises 47 frequently used instructions, offering the following features:

(1) Single-step memory-to-memory operations

(2) Binary operations and decimal operations

(3) Table referencing in program memory (ROM)

(4) Ability to perform branching and subroutine calls using a register value as the address

(5) Systematic set comprising 47 instructions

This manual describes 17K series instructions. Since the manual includes instructions which cannot be used with certain products and instructions with restricted use, the Data Sheet for the product used should always be consulted when programming is carried out.

1.2 INSTRUCTION CONFIGURATION

17K series operation codes are of three kinds:

(1) Zero-operand instructions

These are instructions such as "INC AR", "PUSH AR", "RET", etc., in which the operand is fixed or there is no operand.

(2) Single-operand instructions

Instructions such as "RORC r", "STOP s", etc., in which an address or immediate data is written as the operand.

(3) Two-operand instructions

Instructions such as "ADD r, m", "ADD m, #i", etc., in which two addresses or an address and immediate data are written as the operands.

CHAPTER 2. PROGRAM MEMORY (ROM)

Store the "program" which the CPU executes and the predetermined "constant data" in the program memory (ROM).

2.1 PROGRAM MEMORY CONFIGURATION

Figure 2-1 shows the program memory configuration.

As shown in Figure 2-1, the program memory consists of a maximum of 64K steps x 16 bits.

"Addresses" are allocated in 16-bit units in the program memory ranging from 0000H to FFFFH. In addition, the program memory is divided into segments of 8K steps and each segment is divided into pages of 2K steps.

One of the segments is a special segment called "system segment". Which segment is designated as the system segment depends on the product. There may be no system segment in certain products.

Figure 2-2 shows the system segment configuration. As shown in this figure, system segment entry addresses are allocated in system segment page 0.

Figure 2-1 Program Memory Configuration

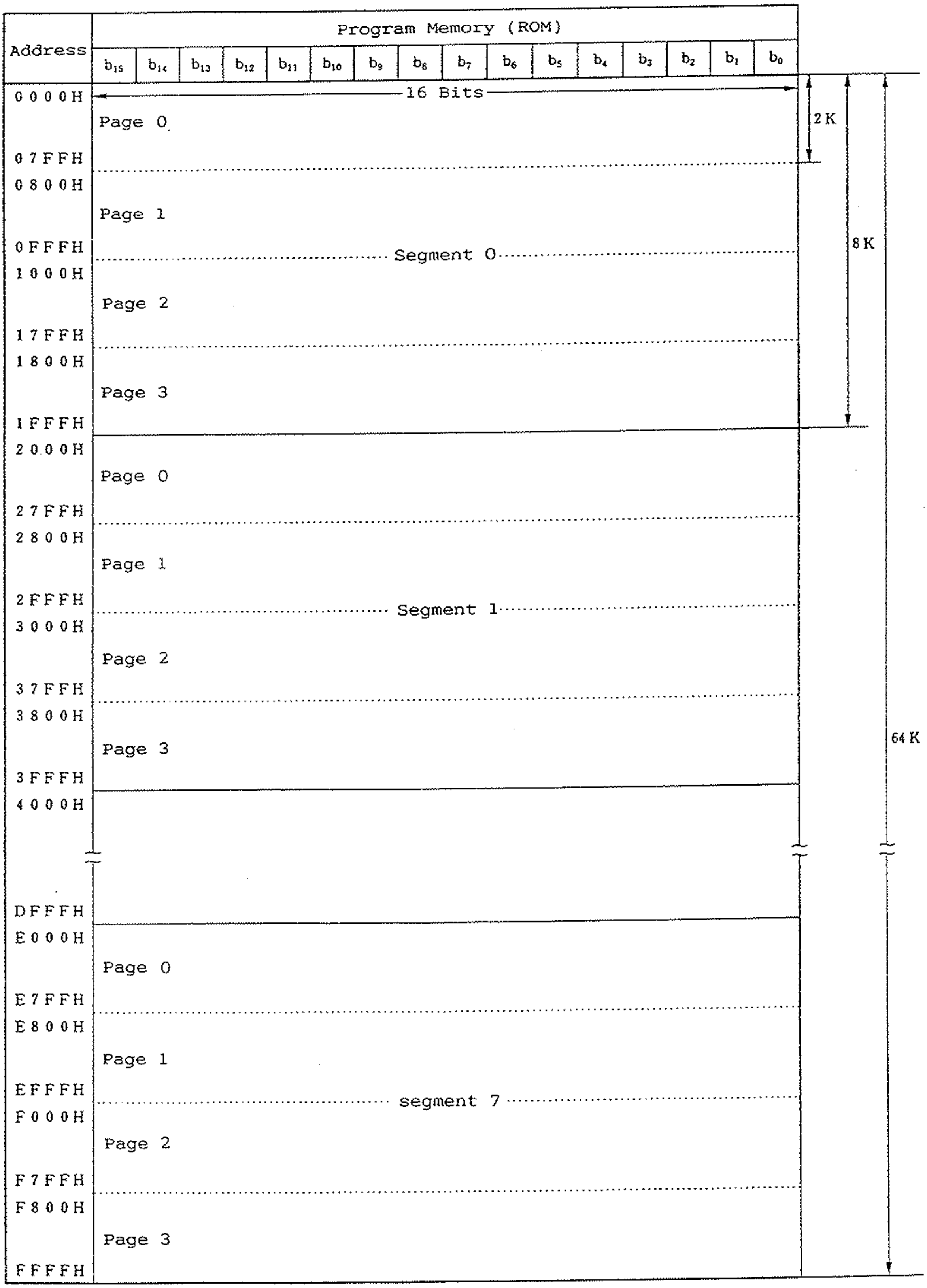
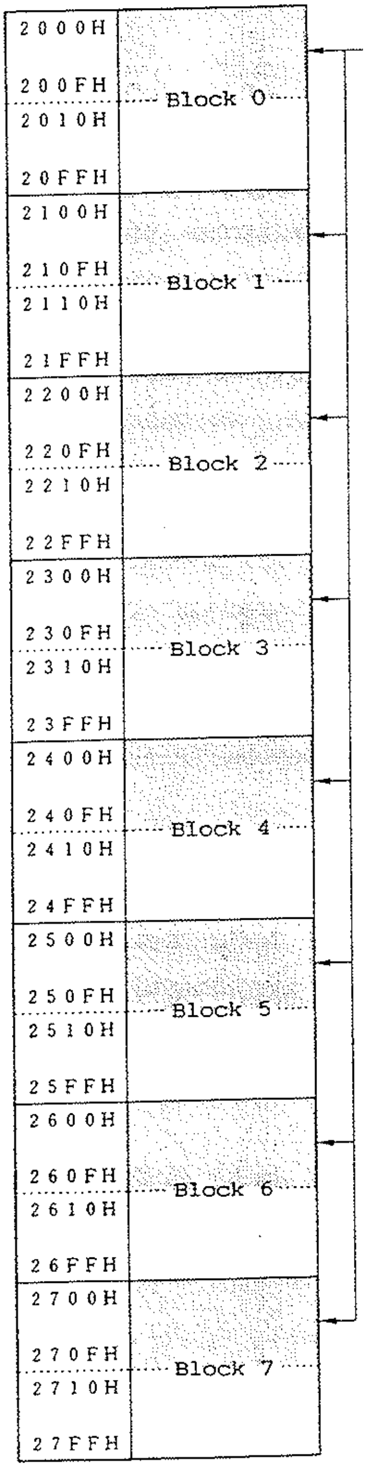


Figure 2-2 System Segment Configuration (Page 0)

When the system segment is allocated in segment 1



System Segment Entry Address
(▨ Part) 16 Steps x 8

2.2 PROGRAM MEMORY FUNCTIONS

Program memory has two major functions.

(1) Storing program
(2) Storing constant data

The program consists of a group of instructions which operate the CPU. The CPU performs processing sequentially in accordance with the instructions written in the program. That is, the CPU reads sequentially the instructions from the program stored in the program memory and performs processing according to each instruction.

Since every "instruction" is a 16-bit "one-word" instruction, one instruction can be stored at one program memory address.

The constant data is the predetermined data such as display pattern.

The constant data can read data from the program memory to the data buffer (DBF) on the data memory (RAM) by using the "MOVT DBF, @AR" special instruction. Reading the constant data on the program memory in such a way is called "table reference".

The program memory is a read only memory and cannot be rewritten by an "instruction". Therefore, the program memory and ROM (Read Only Memory) are used as having the same meaning.

2.3 PROGRAM MEMORY FLOW

The program stored in the program memory is normally executed one address at a time starting from 0000H. However, if, for example, a different program is executed under a certain condition, it is necessary to branch the program flow. In such a case, a branch instruction (BR) is used.

When the same program is to be executed repeatedly, if the same program is used every time, the program memory efficiency drops. In that case, the same program can be executed by calling it by the "CALL" and "SYSCAL" special instructions which were placed in the same location in advance. This program is called "subroutine".

In contrast to the subroutine, the program which is normally executed is called "main routine".

When the program is executed one address at a time, if the last address (address 1FFFH in case of segment 0) is executed, the address of the program to be executed next is not the start address (address 20000H in case of segment 0) but the start address (address 0000H in case of segment 0) of the same segment. Therefore, ensure that a branch instruction is described at the last address of each segment. When the program with different segment is executed, an indirect branch instruction or indirect subroutine call instruction or system subroutine call instruction is used.

If there is a program which is to be executed when a certain condition is established irrespective of the program flow, an interrupt function is used. When the condition is met, the interrupt function can be branched to the predetermined address (interrupt vector address) irrespective of the current program flow.

The program flow described above is controlled by the program counter (PC) which specifies the program memory address.

CHAPTER 3. PROGRAM COUNTER (PC)

The program counter (PC) is used to specify program memory addresses.

3.1 PROGRAM COUNTER CONFIGURATION

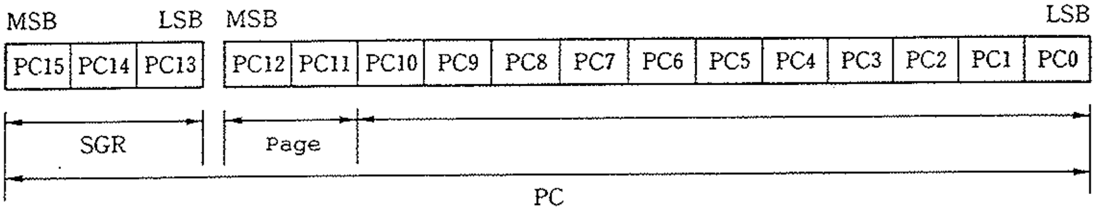
The program counter, as shown in Figure 3-1, consists of a total of 16 bits: a 13-bit binary counter and a 3-bit segment register (SGR).

The 13-bit binary counter is incremented each time an instruction is executed. However, no carry from 13-bit binary counter to the segment register is performed.

The program counter performs a data transfer between the address stack and the address register in 16-bit units.

At this time, those bits of the program counter which are outside the program memory address range are fixed at 0.

Figure 3-1 Program Counter Configuration



3.2 PROGRAM COUNTER OPERATION

Normally, the program counter is automatically incremented each time an instruction is executed. When a branch instruction, subroutine call instruction, system subroutine call instruction, return instruction or table reference instruction is executed, or when an interrupt is acknowledged, or after a reset, the values specified for the program counter are set.

The program counter values after each instruction is executed are shown in Figure 3-2.

Figure 3-2 Program Counter Values after Instruction is Executed

Program Counter / Instructions		Program Counter (PC) Contents															
		PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
BR addr	Page 0	Invariable			0	0	Instruction operand (addr)										
	Page 1				0	1											
	Page 2				1	0											
	Page 3				1	1											
CALL addr		Invariable			0	0	Instruction operand (addr)										
SYSCAL entry		System segment			0	0	Instruction operand entry$_H$			0	0	0	0	Instruction operand entry$_L$			
BR @AR CALL @AR MOVT DBF, @AR		Address register contents															
RET RETSK RETI MOVT DBF, @AR		Address stack register (ASR) contents specified by stack pointer (SP) (return address)															
Interrupt acknowledgment		Interrupt vector address															
After reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHAPTER 4. DATA MEMORY ADDRESS SPECIFICATION METHODS

A data memory address consists of 3 items: Bank (4 bits), row address (3 bits), and column address (4 bits).

In this manual, an address written as "0.43H", for example, indicates bank 0, row address 4, and column address 3.

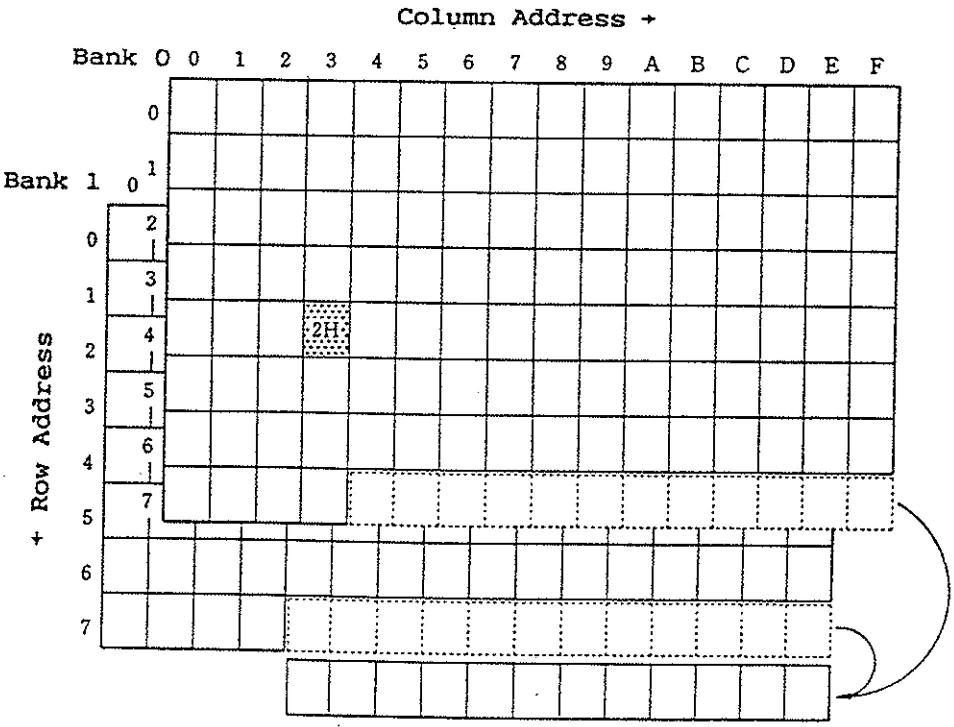
4.1 DATA MEMORY DIRECT ADDRESSING

When directly specifying data memory, the bank is specified by the BANK system register (bank register: 79H), and the row address and column address are specified by the instruction operand m (7 bits).

Example:

When BANK = 0

```
MEM043  MEM  0.43H
        MOV  MEM043, #2H
                |      |
                |      └──Immediate data
                |
                └──m:  Address (column address = 3, row
                       address = 4)
```



74H to 7FH are system registers irrespective of the bank specification.

4.2 GENERAL REGISTER ADDRESSING

When a general register is specified, the bank and row address are specified by the RPH and RPL system registers (register pointer: 7DH, 7EH), and the column address is specified by the instruction operand r (4 bits).

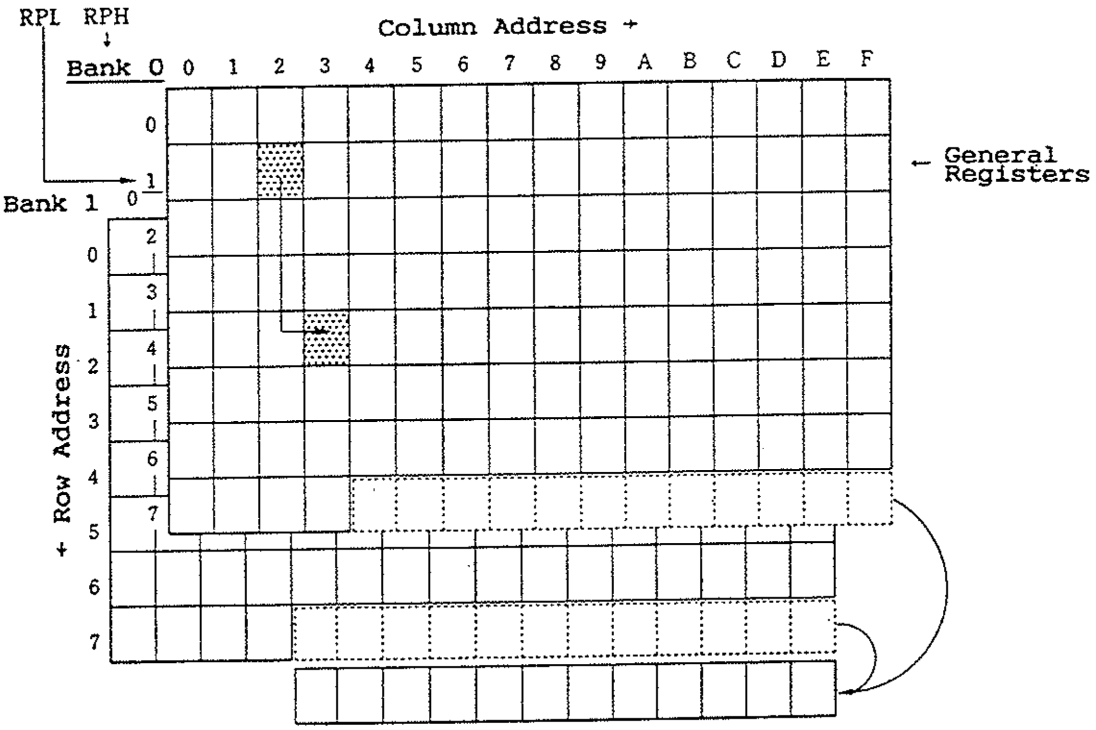
Example:

When BANK = 0, RPH = 0, RPL = 2 (RP = 0000001)

```
MEM002   MEM   0.02H
MEM043   MEM   0.43H
         ST    MEM043, MEM002
```

r: General register column address (0.02H)

Address (0.43H)



74H to 7FH are system registers irrespective of the bank specification.

4.3 DATA MEMORY INDEX MODIFICATION ADDRESSING

When system register IXE (index enable flag: 7FH.0) is set to "1", the address specified by the BANK system register (bank register: 79H) and the instruction operand m (7 bits) is ORed with the contents of system registers IXH, IXM and IXL (index registers: 7AH, 7BH, 7CH), and the result is used as the specified data memory address.

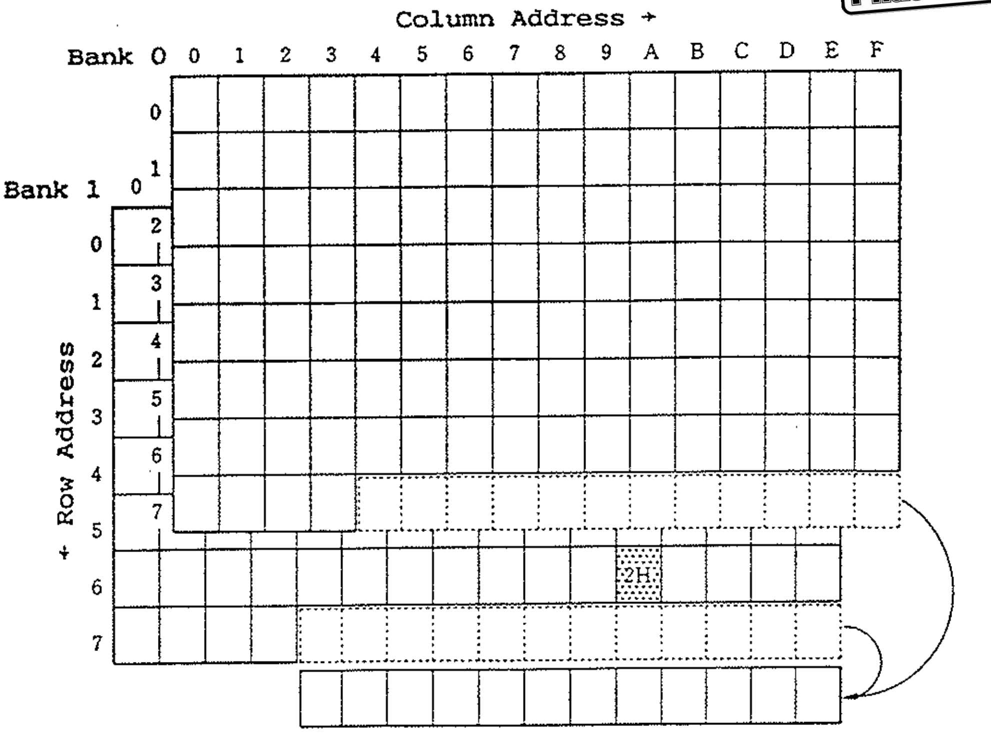
Therefore, when performing data manipulation in memory, when IXE is set to "1" and the start address of the area to be manipulated is set by IXH, IXM and IXL, a method similar to relative addressing can be used, facilitating address modification in the program.

Example:

When BANK = 0, IXE = 1, IXH = 0, IXM = 0EH, IXL = 8

```
 MEM043  MEM  0.43H
         MOV  MEM043, #2H
                ______  ___
                  |      |
                  |      └── Immediate data
                  |
                  └── m:  Address (0.43H)
```

```
Data memory address = [BANK, m] OR [IXH, IXM, IXL]
                    = [0000 1000011B] OR [000 1110 1000B]
                    = [0001 1101011B]
                    = 6BH in bank 1
```



74H to 7FH are system registers irrespective of the bank specification.

4.4 DATA MEMORY GENERAL REGISTER INDIRECT ADDRESSING

The data memory address specification method used when executing a "MOV @r, m" or "MOV m, @r" general register indirect transfer instruction is described below.

In memory indirect addressing, MPE (the memory pointer enable flag) must be set in addition to IXE.

(1) MPE = 0, IXE = 0

When MPE = 0, an indirect transfer is performed between locations with the same row address in the same bank. The bank for direct specification by the operand m is specified by the BANK system register (bank register: 79H), and the row address and column address are specified by the instruction operand m (7 bits).

The bank for indirect specification by the operand @r is specified by the BANK system register (bank register: 79H), and the row address is specified by the high-order 3 bits of the operand m. The column address is specified by the value of a general register. In this case, the general register bank and row address are specified by the RPH and RPL system registers (register pointer: 7DH, 7EH), and the column address is specified by the instruction operand r (4 bits).

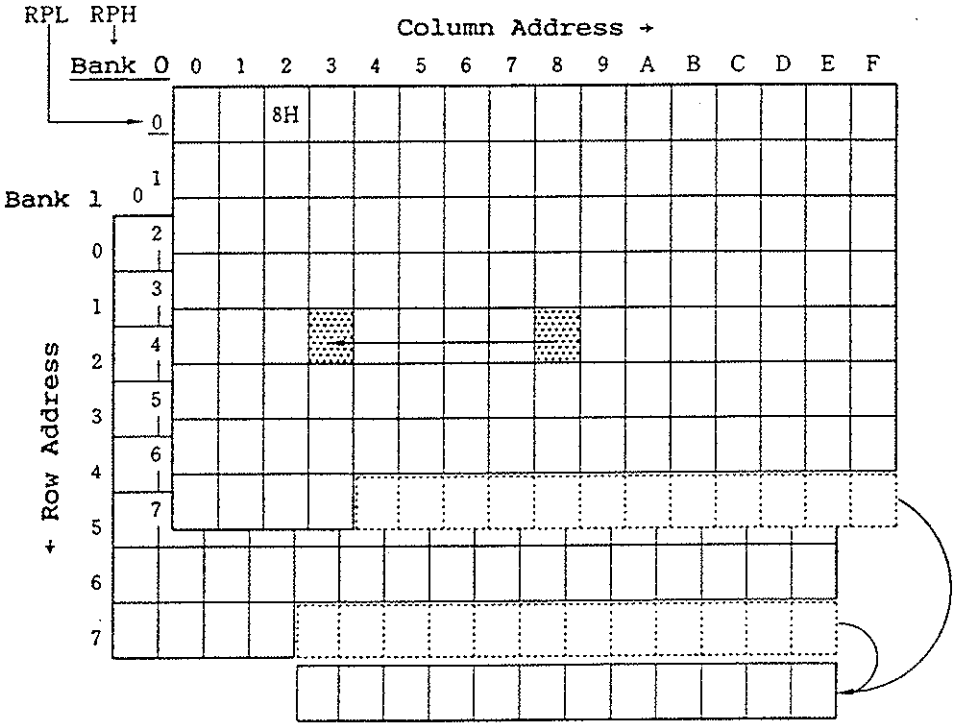
Example:

When BANK = 0, RPH = 0, RPL = 0, and the value of address 0.02H is 8H

```
MEM043  MEM  0.43H
MEM002  MEM  0.02H
        MOV  MEM043, @MEM002
               |        |
               |        └──r:  Indirect
               |               specification
               |               address (0.02H)
               |
               └──m:  Direct specification address
                      (0.43H)
```

Direct specification address = [BANK, m]
= [0000 1000011B]
= 43H in bank 0

Indirect specification address = [BANK, m_H, (R)]
= [0000 100 1000B]
= 48H in bank 0



74H to 7FH are system registers irrespective of the bank specification.

(2) MPE = 1, IXE = 0

When MPE = 1, an indirect transfer can be performed between any data memory locations.

The bank for direct specification by the operand m is specified by the BANK system register (bank register: 79H), and the row address and column address are specified by the instruction operand m (7 bits).

The bank and row address for indirect specification by the operand @r are specified by the MPH and MPL system registers (memory pointer: 7AH, 7BH), and the column address is specified by the value of a general register. In this case, the general register bank and row address are specified by the RPH and RPL system registers (register pointer: 7DH, 7EH), and the column address is specified by the instruction operand r (4 bits).

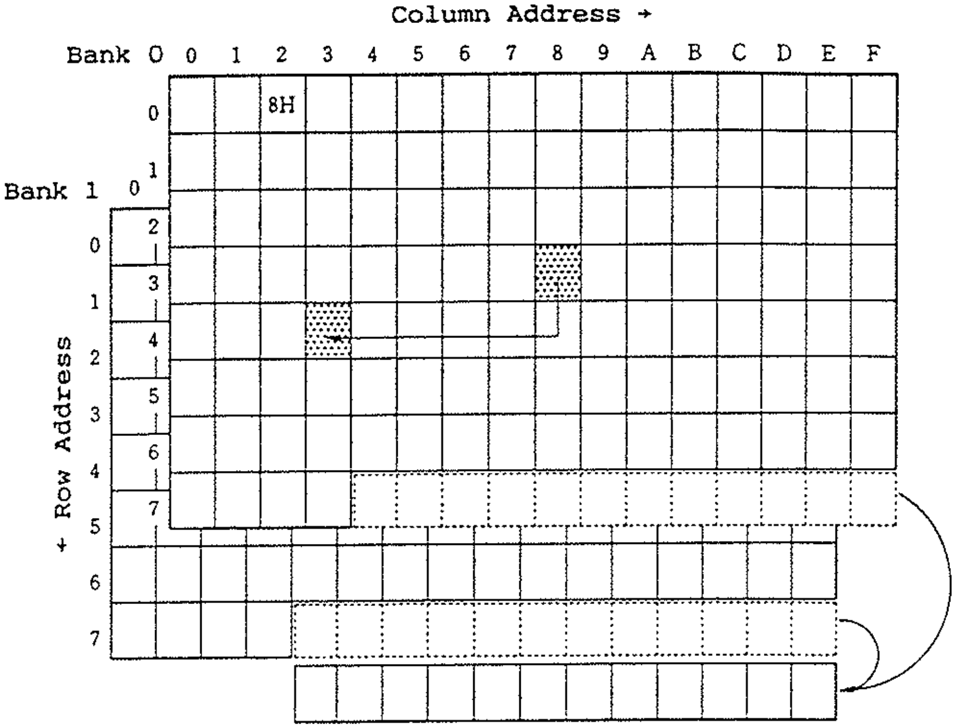
Example:

When BANK = 0, MPH = 0, MPL = 3, RPH = 0, RPL = 0, and the value of address 0.02H is 8H

```
MEM043  MEM  0.43H
MEM002  MEM  0.02H
        MOV  MEM043, @MEM002
                |          └─ r:  Indirect
                |                 specification
                |                 address (0.02H)
                └─m:  Direct specification address
                      (0.43H)
```

```
Direct specification address   = [BANK, m]
                               = [0000 1000011B]
                               = 43H in bank 0
```

Indirect specification address = [MPH, MPL, (R)]
= [000 0011 1000B]
= 38H in bank 0



74H to 7FH are system registers irrespective of the bank specification.

(3) MPE = 0, IXE = 1

When MPE = 0, an indirect transfer is performed between locations with the same row address in the same bank.

For the bank, row address and column address for direct specification by the operand m, the data memory address specified is the result of ORing the address specified by the BANK system register (bank register: 79H) and the instruction operand m (7 bits) with the contents of the IXH, IXM and IXL system registers (index registers: 7AH, 7BH, 7CH).

The bank and row address for indirect specification by the operand @r are specified by the result of ORing the address specified by the BANK system register (bank register: 79H) and the high-order 3 bits of the instruction operand m with the contents of the IXH and IXM system registers (index registers: 7AH, 7BH), and the column address is specified by the value of a general register. In this case, the general register bank and row address are specified by the RPH and RPL system registers (register pointer: 7DH, 7EH), and the column address is specified by the instruction operand r (4 bits).

Example:

When BANK = 0, IXH = 0, IXM = 2, IXL = 4, RPH = 0, RPL = 0, and the value of address 0.02H is 8H

```
MEM043  MEM  0.43H
MEM002  MEM  0.02H
        MOV  MEM043, @MEM002
               │         └─r:  Indirect
               │               specification
               │               address (0.02H)
               └─m:  Direct specification address
                     (0.43H)
```

Direct specification address

= [BANK, m] OR [IXH, IXM, IXL]

= [0000 1000011B] OR [000 0010 0100B]

= [0000 1100111B]

= 67H in bank 0

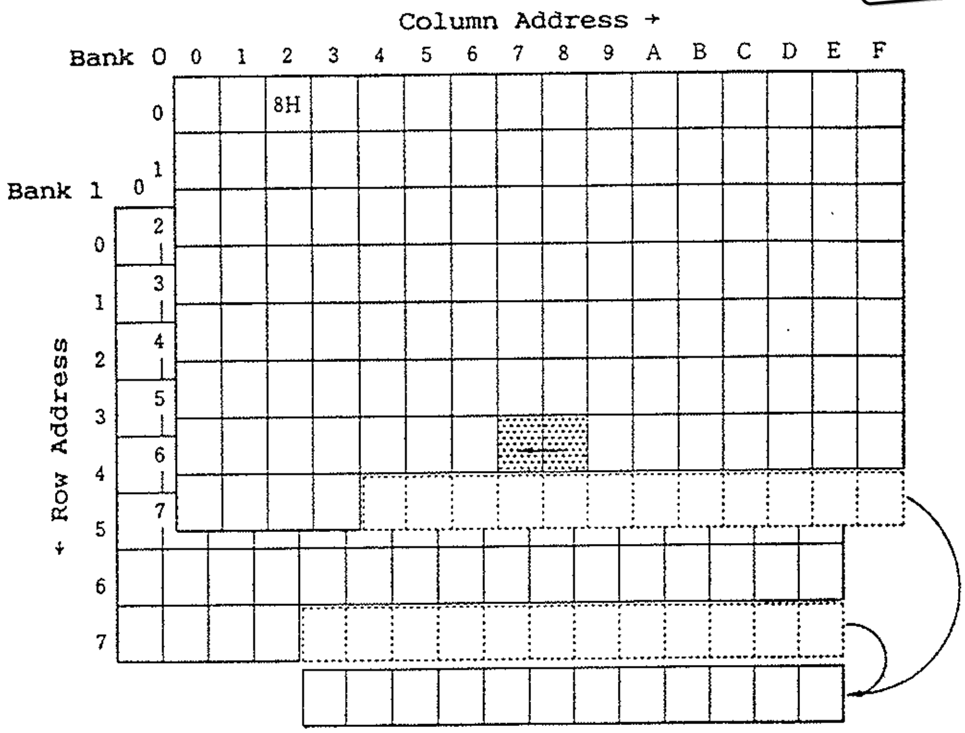
Indirect specification address

= [BANK, m_H, (R)] OR [IXH, IXM, 0]

= [0000 100 1000B] OR [000 0010 0000B]

= [0000 110 1000B]

= 68H in bank 0



74H to 7FH are system registers irrespective of the bank specification.

(4) MPE = 1, IXE = 1

When MPE = 1, an indirect transfer can be performed between any data memory locations.

For the bank, row address and column address for direct specification by the operand m, the data memory address specified is the result of ORing the address specified by the BANK system register (bank register: 79H) and the instruction operand m (7 bits) with the contents of the IXH, IXM and IXL system registers (index registers: 7AH, 7BH, 7CH).

The bank and row address for indirect specification by the operand @r are specified by the MPH and MPL system registers (memory pointer: 7AH, 7BH), and the column address is specified by the value of a general register. In this case, the general register bank and row address are specified by the RPH and RPL system registers (register pointer: 7DH, 7EH), and the column address is specified by the instruction operand r (4 bits).

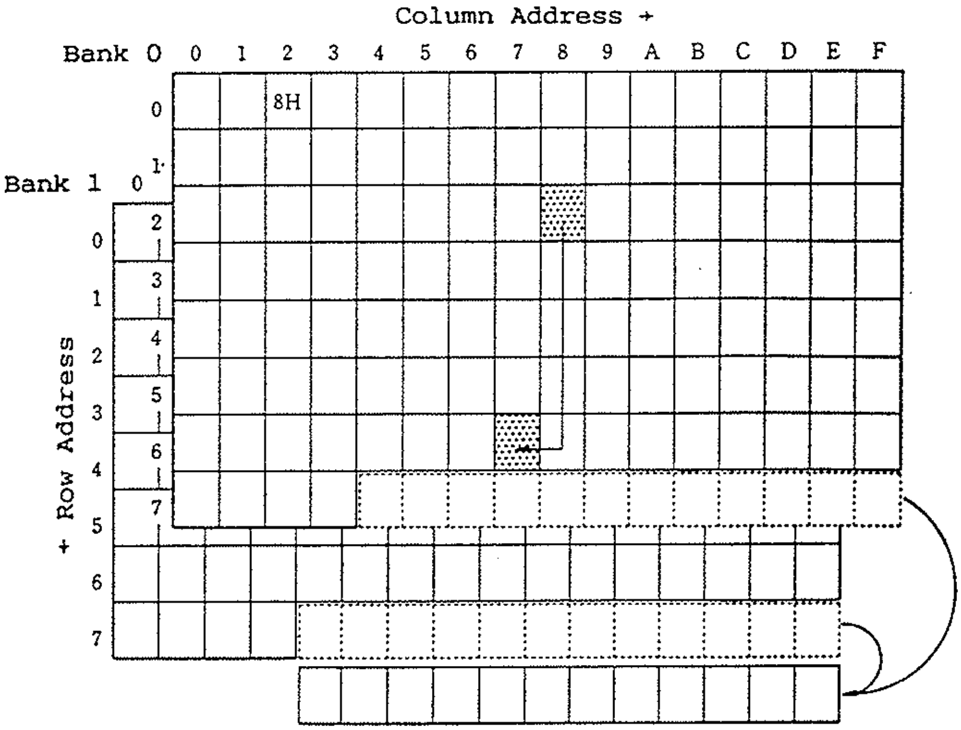
Example:

When BANK = 0, IXH (MPH) = 0, IXM (MPL) = 2, IXL = 4, RPH = 0, RPL = 0, and the value of address 0.02H is 8H

```
MEM043  MEM  0.43H
MEM002  MEM  0.02H
        MOV  MEM043, @MEM002
               |         └─r:  Indirect
               |               specification
               |               address (0.02H)
               └─m:  Direct specification address
                     (0.43H)
```

Direct specification address
= [BANK, m] OR [IXH, IXM, IXL]
= [0000 1000011B] OR [000 0010 0100B]
= [0000 1100111B]
= 67H in bank 0

Indirect specification address
= [MPH, MPL, (R)]
= [0000 0010 1000B]
= 28H in bank 0



74H to 7FH are system registers irrespective of the bank specification.

CHAPTER 5. INSTRUCTION SET

This chapter covers the instruction set. The symbols used in explanation of the instruction set are detailed below.

Legend

AR	:	Address register
ASR	:	Address stack register indicated by stack pointer
(AR) rom	:	program memory data indicated by address register
addr	:	Program memory address (low-order 11 bits)
BANK	:	Bank register
CMP	:	Compare flag
CY	:	Carry flag
DBF	:	Data buffer
entry	:	Program memory address (bits 10 to 8, bits 3 to 0)
entry_H	:	Program memory address (bits 10 to 8)
entry_L	:	Program memory address (bits 3 to 0)
h	:	Halt release condition
INTEF	:	Interrupt enable flag
INTR	:	Register saved to interrupt stack
INTSK	:	Interrupt stack register
IX	:	Index register
IXE	:	Index enable flag
i	:	Immediate data (4 bits)
M	:	Data memory address
		When IXE = 0, M = [(BANK), m_H, m_L]
		When IXE = 1, M = [(BANK), m_H, m_L] OR (IX)
m	:	Data memory address excluding bank
m_H	:	Data memory row address (3 bits)
m_L	:	Data memory column address (4 bits)
MP	:	Data memory row address pointer
MPE	:	Memory pointer enable flag
n	:	Bit position (4 bits)
PAGE	:	Page (program counter bits 12 and 11)
PC	:	Program counter
PE	:	Peripheral register
p	:	Peripheral address

p_H	:	Peripheral address (high-order 3 bits)
p_L	:	Peripheral address (low-order 4 bits)
R	:	General register address R = [(RP), r]
r	:	General register column address
RP	:	General register pointer
RF	:	Register file
rf	:	Register file address
rf_H	:	Register file address (high-order 3 bits)
rf_L	:	Register file address (low-order 4 bits)
SGR	:	Segment register (program counter bits 15 to 13)
SP	:	Stack pointer
SYSSEG	:	System segment address
s	:	Stop release condition
WR	:	Window register
[ ]	:	Data memory or register address
( )	:	Data memory or register value

NOTE: In the following text, unless otherwise specified the settings of the main registers are as follows:

BANK = 0
RPH = 0, RPL = 0
IXE = 0

5.1 ADD r, m — Add data memory to general register

① Operation code

00000	m_H	m_L	r

② Function

When CMP = 0: R, CY ← (R) + (M)

Adds the contents of the data memory addressed by M to the contents of the general register indicated by R, and stores the result in the general register indicated by R.

When CMP = 1: CY ← (R) + (M)

The result is not stored in the register, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a carry is generated as a result of the addition the carry flag (CY) is set; if no carry is generated the carry flag (CY) is reset.

If the result of the addition is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the addition is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the addition is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of addition, binary and BCD operations; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To add the contents of address 0.2FH to the contents of address 0.03H and store the result of the addition in address 0.03H when bank 0 row address 0 (0.00H to 0.0FH) is specified as general registers (RPH = 0, RPL = 0).

```
            0.03H ← (0.03H) + (0.2FH)
MEM003  MEM  0.03H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H ; Data memory bank set to 0
        MOV  RPH, #00H  ; General register bank set to 0
        MOV  RPL, #00H  ; General register row address set to 0
        ADD  MEM003, MEM02F
```

Example 2

To add the contents of address 0.2FH to the contents of address 1.23H and store the result of the addition in address 1.23H when bank 1 row address 2 (1.20H to 1.2FH) is specified as general registers (RPH = 1, RPL = 4).

```
          1.23H ← (1.23H) + (0.2FH)
MEM123    MEM  1.23H
MEM02F    MEM  0.2FH
          MOV  BANK, #00H ; Data memory bank set to 0
          MOV  RPH, #01H  ; General register bank set to 1*
          MOV  RPL, #04H  ; General register row address set to 2
          ADD  MEM123, MEM02F
```

*

Reg-ister	RP							
	RPH				RPL			
Bits	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data	Bank				Row address			B C D

The RP (general register pointer) allocation in the system register is shown in the figure above.

Therefore, to set bank 1 row address 2 for the general registers, 01H must be stored in RPH and 04H in RPL.

Since the BCD flag is reset in this case, subsequent arithmetic operations are hexadecimal operations.

Example 3

To add the contents of address 0.6FH to the contents of address 0.03H and store the result of the addition in address 0.03H. If IXE = 1, IXH = 0, IXM = 4 and IXL = 0, that is IX = 0.40H, data memory 0.6FH can be specified by making the data memory address 2FH.

```
    0.03H ← (0.03H) + (0.6FH)
                       └─ Address obtained by ORing index
                          register contents 0.40H and data
                          memory address 0.2FH

MEM003  MEM  0.03H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H       ; Data memory bank set to 0
        MOV  RPH, #00H        ; General register bank set to 0
        MOV  RPL, #00H        ; General register row address set to 0
        MOV  IXH, #00H        ; IX ←00001000000B
        MOV  IXM, #04H        ;
        MOV  IXL, #00H        ;
        SET1 IXE              ; IXE flag ← 1
        ADD  MEM003, MEM02F   ; IX                     00001000000B(0.40H)
                              ; Bank operand        OR)00000101111B(0.2FH)
                              ; Specified address      00001101111B(0.6FH)
```

Example 4

To add the contents of address 2.3FH to the contents of address 0.03H and store the result of the addition in address 0.03H. If IXE = 1, IXH = 1, IXM = 1 and IXL = 0, that is IX = 2.10H, data memory 2.3FH can be specified by making the data memory address 2FH.

```
0.03H ← (0.03H) + (2.3FH)
                    └ Address obtained by ORing index register
                      contents 2.10H and data memory
                      address 0.2FH
```

```
MEM003  MEM   0.03H
MEM22F  MEM   2.2FH
        MOV   BANK, #00H
        MOV   RPH, #00H       ; General register bank set to 0
        MOV   RPL, #00H       ; General register row address set to 0
        MOV   IXH, #01H       ; IX ← 00100010000B (2.10H)*
        MOV   IXM, #01H
        MOV   IXL, #00H
        SET1  IXE             ; IXE flag ← 1
        ADD   MEM003, MEM22F ; IX                  00100010000B(2.10H)
                             ; Bank operand     OR) 00000101111B(0.2FH)
                             ; Specified            00100111111B(2.3FH)
                               address
```

*

Register	IX											
	IXH				IXM				IXL			
Bits	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data	M P E	← Bank			→	← Row address		→	← Column address			→

The IX (index register) allocation in the system register is shown in the figure above.

Therefore, to set IX = 2.10H, 01H must be stored in IXH, 01H in IXM, and 00H in IXL.

Since the MPE (memory pointer enable) flag is reset in this case, the MP (memory pointer) in a general register indirect transfer is invalid.

④ Note

The first operand of the "ADD r, m" instruction is the general register column address. Thus, when the following coding is used, the general register column address is 03H.

```
MEM013  MEM  0.13H
MEM02F  MEM  0.2FH
        ADD  MEM013, MEM02F
                └─Indicates the general register
                  column address:  The low-order 4
                  bits (03H in this case) are
                  valid.
```

When the CMP flag = 1, the result of the addition is not stored.

When the BCD flag = 1, the result of a decimal addition is stored.

5.2 ADD m, #i Add immediate data to data memory

① Operation code

10000	m_H	m_L	i

② Function

When CMP = 0: M, CY ← (M) + i

Adds the immediate data i to the contents of the data memory addressed by M, and stores the result in the data memory addressed by M.

When CMP = 1: CY ← (M) + i

The result is not stored in the data memory, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a carry is generated as a result of the addition the carry flag (CY) is set; if no carry is generated the carry flag (CY) is reset.

If the result of the addition is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the addition is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the addition is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of addition, binary and BCD; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To add 5 to the contents of address 0.2FH and store the result in address 0.2FH.

```
                0.2FH ← (0.2FH) +05H
 MEM02F  MEM  0.2FH
         MOV  BANK, #00H       ; Data memory bank set to 0
         ADD  MEM02F, #05H
```

Example 2

To add 5 to the contents of address 0.6FH and store the result in address 0.6FH. If IXE = 1, IXH = 0, IXM = 4 and IXL = 0, that is IX = 0.40H, data memory 0.6FH can be specified by making the data memory address 2FH.

```
0.6FH ← (0.6FH) +05H
          └ Address obtained by ORing index register
            contents 0.40H and data memory address 0.2FH
```

```
MEM02F  MEM  0.2FH
        MOV  BANK, #00H     ; Data memory bank set to 0
        MOV  IXH, #00H      ; IX ←00001000000B (0.40H)
        MOV  IXM, #04H
        MOV  IXL, #00H
        SET1 IXE            ; IXE flag ← 1
        ADD  MEM02F, #05H   ; IX                 00001000000B(0.40H)
                            ; Bank operand    OR)00000101111B(0.2FH)
                            ; Specified          00001101111B(0.6FH)
                              address
```

Example 3

To add 5 to the contents of address 2.2FH and store the result in address 2.2FH. If IXE = 1, IXH = 1, IXM = 0 and IXL = 0, that is IX = 2.00H, data memory 2.2FH can be specified by making the data memory address 2FH.

```
2.2FH ← (2.2FH) +05H
          └ Address obtained by ORing index register
            contents 2.00H and data memory address 0.2FH
```

```
MEM22F  MEM   2.2FH
        MOV   BANK, #00H     ; Data memory bank set to 0
        MOV   IXH, #01H      ; IX ←00100000000B
        MOV   IXM, #00H
        MOV   IXL, #00H
        SET1  IXE            ; IXE flag ← 1
        ADD   MEM22F, #05H   ; IX                 00100000000B(2.00H)
                             ; Bank operand    OR)00000101111B(0.2FH)
                             ; Specified          00100101111B(2.2FH)
                               address
```

④ Note

When the CMP flag = 1, the result of the addition is not stored.

When the BCD flag = 1, the result of a decimal addition is stored.

5.3 ADDC r, m Add data memory to general register with carry flag

① Operation code

00010	m_H	m_L	r

② Function

When CMP = 0: R, CY ← (R) + (M) + (CY)

Adds the contents of the data memory addressed by M and the value of the carry flag (CY) to the contents of the general register indicated by R, and stores the result in the general register indicated by R.

When CMP = 1: CY ← (R) + (M) + (CY)

The result is not stored in the register, but the carry flag (CY) and zero flag (Z) are changed according to the result.

Use of this "ADDC" instruction facilitates the addition of two or more words.

If a carry is generated as a result of the addition the carry flag (CY) is set; if no carry is generated the carry flag (CY) is reset.

If the result of the addition is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the addition is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the addition is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of addition, binary and BCD operations; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To add the contents of the 12 bits from address 0.2DH to address 0.2FH to the contents of the 12 bits from address 0.0DH to address 0.0FH and store the result in the 12 bits from address 0.0DH to address 0.0FH when bank 0 row address 0 (0.00H to 0.0FH) is specified as general registers.

$$\mathrm{0.0FH} \leftarrow (\underline{\mathrm{0.0FH}}) + (\mathrm{0.2FH})$$

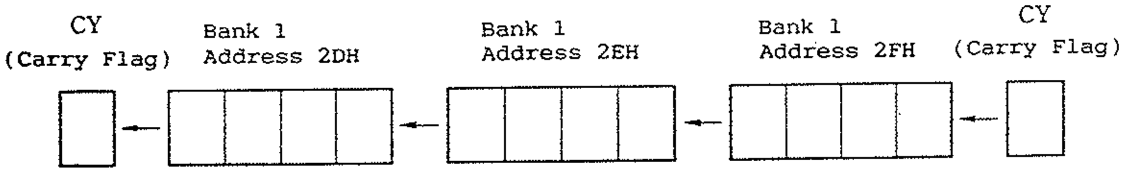
$$\mathrm{0.0EH} \leftarrow (\underline{\mathrm{0.0EH}}) + (\mathrm{0.2EH}) + \mathrm{CY}$$

$$\mathrm{0.0DH} \leftarrow (\underline{\mathrm{0.0DH}}) + (\mathrm{0.2DH}) + \mathrm{CY}$$

```
MEM00D  MEM  0.0DH
MEM00E  MEM  0.0EH
MEM00F  MEM  0.0FH
MEM02D  MEM  0.2DH
MEM02E  MEM  0.2EH
MEM02F  MEM  0.2FH
        MOV  BANK, #00H; Data memory bank set to 0
        MOV  RPH, #00H  ; General register bank set to 0
        MOV  RPL, #00H  ; General register row address set to 0
        ADD  MEM00F, MEM02F
        ADDC MEM00E, MEM02E
        ADDC MEM00D, MEM02D
```

Example 2

To shift the contents of the 12 bits from address 1.2DH to address 1.2FH including the carry flag one bit to the left when bank 1 row address 2 (1.20H to 1.2FH) is specified as general registers.



```
MEM12D  MEM   1.2DH
MEM12E  MEM   1.2EH
MEM12F  MEM   1.2FH
        MOV   BANK, #01H       ; Data memory bank set to 1
        MOV   RPH, #01H        ; General register bank set to 1
        MOV   RPL, #04H        ; General register row address set to 2
        ADDC  MEM12F, MEM12F
        ADDC  MEM12E, MEM12E
        ADDC  MEM12D, MEM12D
```

Example 3

To add the contents of addresses 0.40H through 0.4FH to the contents of address 0.0FH and store the result in address 0.0FH.

```
            0.0FH ← (0.0FH) + (0.40H) + (0.41H) +……+ (0.4FH)
MEM00F   MEM   0.0FH
MEM000   MEM   0.00H
         MOV   BANK, #00H        ; Data memory bank set to 0
         MOV   RPH, #00H         ; General register bank set to 0
         MOV   RPL, #00H         ; General register row address
                                   set to 0
         MOV   IXH, #00H         ; IX ←00001000000B (0.40H)
         MOV   IXM, #04H
         MOV   IXL, #00H
LOOP1 :
         SET1  IXE               ; IXE flag ← 1
         ADD   MEM00F, MEM000
         CLR1  IXE               ; IXE flag ← 0
         INC   IX                ; IX ←IX+1
         SKE   IXL, #0
         JMP   LOOP1
```

Example 4

To add the contents of the 12 bits from address 0.0DH to address 0.0FH to the contents of the 12 bits from address 1.40H to address 1.42H and store the result in the 12 bits from address 0.0DH to address 0.0FH. At this time, if IXE = 1, IXH = 0, IXM = 0CH, IXL = 0, that is, IX = 1.40H, data memory 1.40H, 1.41H and 1.42H can be specified by setting the data memory address to 0.00H, 0.01H and 0.02H, respectively.

```
0.0FH ← (0.0FH) + (1.42H)
0.0EH ← (0.0EH) + (1.41H) +CY
0.0DH ← (0.0DH) + (1.40H) +CY
                   └─ Address determined by index register
                      contents 1.40H ORed with data memory
                      address 0.02H
```

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H      ; Data memory bank set to 0
        MOV   RPH, #00H       ; General register bank set to 0
        MOV   RPL, #00H       ; General register row address set to 0
        MOV   IXH, #00H       ; IX ←00011000000 (1.40H)
        MOV   IXM, #0CH
        MOV   IXL, #00H
        SET1  IXE             ; IXE flag ← 1
        ADD   MEM00F, MEM002 ; 0.0FH ← (0.0FH) + (1.42H)
        ADDC  MEM00E, MEM001 ; 0.0EH ← (0.0EH) + (1.41H)
        ADDC  MEM00D, MEM000 ; 0.0DH ← (0.0DH) + (1.40H)
```

5.4 ADDC m, #i Add immediate data to data memory with carry flag

① Operation code

10010	m_H	m_L	i

② Function

When CMP = 0: M, CY ← (M) + i + (CY)

Adds the immediate data i and the carry flag (CY) to the contents of the data memory addressed by M, and stores the result in the data memory addressed by M.

When CMP = 1: CY ← (M) + i + (CY)

The result is not stored in the data memory, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a carry is generated as a result of the addition the carry flag (CY) is set; if no carry is generated the carry flag (CY) is reset.

If the result of the addition is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the addition is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the addition is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of addition, binary and BCD operations; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To add 5 to the contents of the 12 bits from address 0.0DH to address 0.0FH and store the result in addresses 0.0DH to 0.0FH.

```
                0.0FH ← (0.0FH) +05H
                0.0EH ← (0.0EH) +CY
                0.0DH ← (0.0DH) +CY
MEM00D  MEM     0.0DH
MEM00E  MEM     0.0EH
MEM00F  MEM     0.0FH
        MOV     BANK, #00H      ; Data memory bank set to 0
        ADD     MEM00F, #05H
        ADDC    MEM00E, #00H
        ADDC    MEM00D, #00H
```

Example 2

To add 5 to the contents of the 12 bits from address 0.4DH to address 0.4FH and store the result in addresses 0.4DH to 0.4FH. At this time, if IXE = 1, IXH = 0, IXM = 4, IXL = 0, that is, IX = 0.40H, data memory 0.4DH, 0.4EH and 0.4FH can be specified by setting the data memory address to 0.0DH, 0.0EH and 0.0FH, respectively.

```
0.4FH ← (0.4FH) +05H
0.4EH ← (0.4EH) +CY
0.4DH ← (0.4DH) +CY
          |
          └─ Address determined by index register contents
             0.40H ORed with data memory address 0.0DH
```

```
MEM00D  MEM    0.0DH
MEM00E  MEM    0.0EH
MEM00F  MEM    0.0FH
        MOV    BANK, #00H      ; Data memory bank set to 0
        MOV    IXH, #00H       ; IX ←00001000000B (0.40 H)
        MOV    IXM, #04H
        MOV    IXL, #00H
        SET1   IXE             ; IXE flag ← 1
        ADD    MEM00F, #5      ; 0.4FH ← (0.4FH) +5H
        ADDC   MEM00E, #0      ; 0.4EH ← (0.4EH) +CY
        ADDC   MEM00D, #0      ; 0.4DH ← (0.4DH) +CY
```

5.5 SUB r, m — Subtract data memory from general register

① Operation code

00001	m_H	m_L	r

② Function

When CMP = 0: R, CY ← (R) - (M)

Subtracts the contents of the data memory addressed by M from the contents of the general register indicated by R, and stores the result in the general register indicated by R.

When CMP = 1: CY ← (R) - (M)

The result is not stored in the register, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a borrow is generated as a result of the subtraction the carry flag (CY) is set; if no borrow is generated the carry flag (CY) is reset.

If the result of the subtraction is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the subtraction is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the subtraction is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of subtraction, binary and BCD operations; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To subtract the contents of address 0.2FH from the contents of address 0.03H and store the result of the subtraction in address 0.03H when bank 0 row address 0 (0.00H to 0.0FH) is specified as general registers (RPH = 0, RPL = 0).

```
        0.03H ← (0.03H) − (0.2FH)
MEM003  MEM  0.03H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H     ; Data memory bank set to 0
        MOV  RPH, #00H      ; General register bank set to 0
        MOV  RPL, #00H      ; General register row address set to 0
        SUB  MEM003, MEM02F
```

Example 2

To subtract the contents of address 0.2FH from the contents of address 1.23H and store the result of the subtraction in address 1.23H when bank 1 row address 2 (1.20H to 1.2FH) is specified as general registers (RPH = 1, RPL = 4).

```
        1.23H ← (1.23H) − (0.2FH)
MEM123  MEM  1.23H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H     ; Data memory bank set to 0
        MOV  RPH, #01H      ; General register bank set to 1
        MOV  RPL, #04H      ; General register row address set to 2
        SUB  MEM123, MEM02F
```

Example 3

To subtract the contents of address 0.6FH from the contents of address 0.03H and store the result of the subtraction in address 0.03H. If IXE = 1, IXH = 0, IXM = 4 and IXL = 0, that is IX = 0.40H, data memory 0.6FH can be specified by making the data memory address 2FH.

```
0.03H ← (0.03H) − (0.6FH)
                    └─ Address determined by index register
                       contents 0.40H ORed with data memory
                       address 0.2FH
```

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        MOV   RPH, #00H        ; General register bank set to 0
        MOV   RPL, #00H        ; General register row address set to 0
        MOV   IXH, #00H        ; IX ←00001000000B (0.40H)
        MOV   IXM, #04H        ;
        MOV   IXL, #00H        ;
        SET1  IXE              ; IXE flag ← 1
        SUB   MEM003, MEM02F   ; IX                    00001000000B(0.40H)
                               ; Bank operand       OR)00000101111B(0.2FH)
                               ; Specified address     00001101111B(0.6FH)
```

Example 4

To subtract the contents of address 2.3FH from the contents of address 0.03H and store the result of the subtraction in address 0.03H. If IXE = 1, IXH = 1, IXM = 1 and IXL = 0, that is IX = 2.10H, data memory 2.3FH can be specified by making the data memory address 2FH.

```
0.03H ← (0.03H) − (2.3FH)
                    └─ Address determined by index register
                       contents 2.10H ORed with data memory
                       address 0.2FH

MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H        ; Data memory bank set to 0
        MOV   RPH, #00H         ; General register bank set to 0
        MOV   RPL, #00H         ; General register row address set to 0
        MOV   IXH, #01H         ; IX ←00100010000B (2.10H)
        MOV   IXM, #01H         ;
        MOV   IXL, #00H         ;
        SET1  IXE               ; IXE flag ← 1
        SUB   MEM003, MEM02F    ; IX                 00100010000B(2.10H)
                                ; Bank operand    OR)00000101111B(0.2FH)
                                ; Specified address  00100111111B(2.3FH)
```

④ Note

The first operand of the "SUB r, m" instruction must be the general register address. Thus, when the following coding is used, address 0.03H is specified as the register.

```
MEM013  MEM   0.13H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H        ; Data memory bank set to 0
        MOV   RPH, #00H         ; General register bank set to 0
        MOV   RPL, #00H         ; General register row address set to 0
        SUB   MEM013, MEM02F
              └─ The general register address must be in the range
                 00H to 0FH (register pointer set to other than row
                 address 1).
```

When the CMP flag = 1, the result of the subtraction is not stored.

When the BCD flag = 1, the result of a decimal operation is stored.

5.6 SUB m, #i Subtract immediate data from data memory

① Operation code

10001	m_H	m_L	i

② Function

When CMP = 0: M, CY ← (M) - i

Subtracts the immediate data i from the contents of the data memory addressed by M, and stores the result in the data memory addressed by M.

When CMP = 1: CY ← (M) - i

The result is not stored in the data memory, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a borrow is generated as a result of the subtraction the carry flag (CY) is set; if no borrow is generated the carry flag (CY) is reset.

If the result of the subtraction is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the subtraction is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the subtraction is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of subtraction, binary and BCD operations; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To subtract 5 from the contents of address 0.2FH and store the result in address 0.2FH.

```
                0.2FH ← (0.2FH) − 05H
  MEM02F  MEM   0.2FH
          MOV   BANK, #00H       ; Data memory bank set to 0
          SUB   MEM02F, #05H
```

Example 2

To subtract 5 from the contents of address 0.6FH and store the result in address 0.6FH. If IXE = 1, IXH = 0, IXM = 4 and IXL = 0, that is IX = 0.40H, data memory 0.6FH can be specified by making the data memory address 2FH.

```
0.6FH ← (0.6FH) − 05H
          └─ Address obtained by ORing index register
             contents 0.40H and data memory address 0.2FH
```

```
MEM02F  MEM   0.2FH
        MOV   BANK, #00H     ; Data memory bank set to 0
        MOV   IXH, #00H      ; IX ←00001000000B (0.40H)
        MOV   IXM, #04H      ;
        MOV   IXL, #00H      ;
        SET1  IXE            ; IXE flag ← 1
        SUB   MEM02F, #05H   ; IX                 00001000000B(0.40H)
                             ; Bank operand    OR)00000101111B(0.2FH)
                             ; Specified          00001101111B(0.6FH)
                               address
```

Example 3

To subtract 5 from the contents of address 2.2FH and store the result in address 2.2FH. If IXE = 1, IXH = 1, IXM = 0 and IXL = 0, that is IX = 2.00H, data memory 2.2FH can be specified by making the data memory address 2FH.

$$2.2FH \leftarrow (\underline{2.2FH}) - 05H$$

Address obtained by ORing index register contents 2.00H and data memory address 0.2FH

```
MEM02F  MEM   0.2FH
        MOV   BANK0, #00H    ; Data memory bank set to 0
        MOV   IXH, #01H      ; IX ←00100000000B (2.00H)
        MOV   IXM, #00H      ;
        MOV   IXL, #00H      ;
        SET1  IXE            ; IXE flag ← 1
        SUB   MEM02F, #05H   ; IX                  00100000000B (2.00H)
                             ; Bank operand     OR)00000101111B (0.2FH)
                             ; Specified           00100101111B (2.2FH)
                               address
```

④ Note

When the CMP flag = 1, the result of the subtraction is not stored.

When the BCD flag = 1, the result of a decimal operation is stored.

5.7 SUBC r, m Subtract data memory from general register with carry flag

① Operation code

00011	m_H	m_L	r

② Function

When CMP = 0: $R, CY \leftarrow (R) - (M) - (CY)$

Subtracts the contents of the data memory addressed by M and the value of the carry flag (CY) from the contents of the general register indicated by R, and stores the result in the general register indicated by R. Use of this SUBC instruction facilitates a subtraction of two or more words.

When CMP = 1: $CY \leftarrow (R) - (M) - (CY)$

The result is not stored in the register, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a borrow is generated as a result of the subtraction the carry flag (CY) is set; if no borrow is generated the carry flag (CY) is reset.

If the result of the subtraction is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the subtraction is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the subtraction is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of subtraction, binary and BCD operations; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To subtract the contents of the 12 bits from address 0.2DH to address 0.2FH from the contents of the 12 bits from address 0.0DH to address 0.0FH and store the result in the 12 bits from address 0.0DH to address 0.0FH when bank 0 row address 0 (0.00H to 0.0FH) is specified as general registers.

```
        0.0FH ← (0.0FH) − (0.2FH)
        0.0EH ← (0.0EH) − (0.2EH) −CY
        0.0DH ← (0.0DH) − (0.2DH) −CY
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
MEM02D  MEM   0.2DH
MEM02E  MEM   0.2EH
MEM02F  MEM   0.2FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        MOV   RPH, #00H        ; General register bank set to 0
        MOV   RPL, #00H        ; General register row address set to 0
        SUB   MEM00F, MEM02F
        SUBC  MEM00E, MEM02E
        SUBC  MEM00D, MEM02D
```

Example 2

To subtract the contents of the 12 bits from address 1.40H to address 1.42H from the contents of the 12 bits from address 0.0DH to address 0.0FH and store the result in the 12 bits from address 0.0DH to address 0.0FH. At this time, if IXE = 1, IXH = 0, IXM = 0CH, IXL = 0, that is, IX = 1.40H, data memory 1.40H, 1.41H and 1.42H can be specified by setting the data memory address to 0.00H, 0.01H and 0.02H, respectively.

$$0.0FH \leftarrow (0.0FH) - (1.42H)$$

$$0.0EH \leftarrow (0.0EH) - (1.41H) - CY$$

$$0.00H \leftarrow (0.00H) + (\underline{1.40H}) - CY$$

Address determined by index register contents 1.40H ORed with data memory address 0.00H

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H         ; Data memory bank set to 0
        MOV   RPH, #00H          ; General register bank set to 0
        MOV   RPL, #00H          ; General register row address set to 0
        MOV   IXH, #00H          ; IX ←00011000000B (1.40H)
        MOV   IXM, #0CH          ;
        MOV   IXL, #00H          ;
        SET1  IXE                ; IXE flag ← 1
        SUB   MEM00F, MEM002 ; 0.0FH ← (0.0FH) − (1.42H)
        SUBC  MEM00E, MEM001 ; 0.0EH ← (0.0EH) − (1.41H)
        SUBC  MEM00D, MEM000 ; 0.0DH ← (0.0DH) − (1.40H)
```

Example 3

To compare the contents of the 16 bits from address 0.00H to address 0.03H with the contents of the 16 bits from address 0.0CH to address 0.0FH, and jump to LAB1 if they are the same or to LAB2 if not the same.

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM003  MEM   0.03H
MEM00C  MEM   0.0CH
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H        ; Data memory bank set to 0
        MOV   RPH, #00H         ; General register bank set to 0
        MOV   RPL, #00H         ; General register row address set to 0
        SET2  CMP, Z            ; CMP flag ← 1, Z flag ← 1
        SUB   MEM000, MEM00C ;} Since CMP flag is set, address contents
        SUBC  MEM001, MEM00D ;} 0.00H to 0.03H not changed
        SUBC  MEM002, MEM00E ;
        SUBC  MEM003, MEM00F ;
        SKF1  Z                 ;} If compared contents are the same,
        BR    LAB1              ;} Z flag = 1; if different, Z flag = 0
        BR    LAB2

                 :
   LAB1 :        :
                 :
   LAB2 :        :
                 :
```

5.8 SUBC m, #i Subtract immediate data from data memory with carry flag

① Operation code

10011	m_H	m_L	i

② Function

When CMP = 0: $M, CY \leftarrow (M) - i - (CY)$

Subtracts the immediate data i and the carry flag (CY) from the contents of the data memory addressed by M, and stores the result in the data memory addressed by M.

When CMP = 1: $CY \leftarrow (M) - i - (CY)$

The result is not stored in the data memory, but the carry flag (CY) and zero flag (Z) are changed according to the result.

If a borrow is generated as a result of the subtraction the carry flag (CY) is set; if no borrow is generated the carry flag (CY) is reset.

If the result of the subtraction is non-zero, the zero flag (Z) is reset without regard to the compare flag (CMP).

If the result of the subtraction is zero when the compare flag is reset (CMP = 0), the zero flag (Z) is set.

If the result of the subtraction is zero when the compare flag is set (CMP = 1), the zero flag (Z) is not changed.

There are two kinds of subtraction, binary and BCD; which kind is performed is specified by the BCD flag (BCD) in the PSW.

③ Example 1

To subtract 5 from the contents of the 12 bits from address 0.0DH to address 0.0FH and store the result in addresses 0.0DH to 0.0FH.

```
                0.0FH ← (0.0FH) −05H
                0.0EH ← (0.0EH) −CY
                0.0DH ← (0.0DH) −CY
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H      ; Data memory bank set to 0
        SUB   MEM00F, #05H
        SUBC  MEM00E, #00H
        SUBC  MEM00D, #00H
```

Example 2

To subtract 5 from the contents of the 12 bits from address 0.4DH to address 0.4FH and store the result in addresses 0.4DH to 0.4FH. At this time, if IXE = 1, IXH = 0, IXM = 4, IXL = 0, that is, IX = 0.40H, data memory 0.4DH, 0.4EH and 0.4FH can be specified by setting the data memory address to 0.0DH, 0.0EH and 0.0FH, respectively.

```
0.4FH ← (0.4FH) −05H
0.4EH ← (0.4EH) −CY
0.4DH ← (0.4DH) −CY
          |
          └─ Address determined by index register contents
             0.40H ORed with data memory address 0.0DH
```

```
MEM00D  MEM   0.0DH
MEM00E  MEM   0.0EH
MEM00F  MEM   0.0FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        MOV   IXH, #00H        ; IX ←00001000000B (0.40H)
        MOV   IXM, #04H        ;
        MOV   IXL, #00H        ;
        SET1  IXE              ; IXE flag ← 1
        MOV   BANK, #00H       ; Data memory bank set to 0
        SUB   MEM00F, #05H     ; (0.4FH) ← (0.4FH) −05H
        SUBC  MEM00E, #00H     ; (0.4EH) ← (0.4EH) −CY
        SUBC  MEM00D, #00H     ; (0.4DH) ← (0.4DH) −CY
```

Example 3

To compare the immediate data 0A3FH with the contents of the 16 bits from address 0.00H to address 0.03H, and jump to LAB1 if they are the same or to LAB2 if not the same.

```
MEM000  MEM   0.00H
MEM001  MEM   0.01H
MEM002  MEM   0.02H
MEM003  MEM   0.03H
        MOV   BANK, #00H     ; Data memory bank set to 0
        SET2  CMP, Z         ; CMP flag ← 1, Z flag ← 1
        SUB   MEM000, #0H    ;} Since CMP flag is set, address contents 0.00H
        SUBC  MEM001, #0AH   ;} to 0.03H not changed
        SUBC  MEM002, #3H    ;
        SUBC  MEM003, #0FH   ;
        SKF1  Z              ;} If compared values are the same,
        BR    LAB1           ;} Z flag = 1; if different, Z flag = 0
        BR    LAB2
              :
    LAB1 :    :
              :
    LAB2 :    :
              :
```

5.9 INC AR Increment address register

① Operation code

00111	000	1001	0000

② Function

$AR \leftarrow (AR) + 1$

Increments the address register (AR).

③ Example 1

To add 1 to the contents of the 16 bits from AR3 to AR0 (address register) in the system register and store the result in AR3 to AR0.

```
 AR0 ← AR0 + 1
 AR1 ← AR1 + CY
 AR2 ← AR2 + CY
 AR3 ← AR3 + CY
INC AR
```

This operation can also be performed using add instructions, as follows:

```
ADD   AR0, #01H
ADDC  AR1, #00H
ADDC  AR2, #00H
ADDC  AR3, #00H
```

Example 2

To transfer table data 16 bits (one address) at a time to the DBF (data buffer) using the table reference instruction (see the description of the MOVT instruction in 5.26 for details).

```
; Address            Table data
010H    DW           0F3FFH
011H    DW           0A123H
012H    DW           0FFF1H
013H    DW           0FFF5H
014H    DW           0FF11H
         :
         :
        MOV          AR3, #0H          ; Table data address
        MOV          AR2, #0H          ;} Set 0010H in address
        MOV          AR1, #1H          ;} register.
        MOV          AR0, #0H
LOOP :
        MOVT         DBF, @AR          ;} Table data read in DBF
         :                             ;}
         :
         :                             ; Processing which references
         :                             ; table data
         :
        INC          AR                ;} Address register incremented
        BR           LOOP              ;} by 1
```

④ Note

The number of bits which can be used in the address register (AR3 to AR0) varies from product to product: Please refer to the Data Sheet for the product concerned when using this register.

5.10 INC IX Increment index register

① Operation code

00111	000	1000	0000

② Function

IX ← (IX) + 1

Increments the index register (IX).

③ Example 1

To add 1 to the contents of the 11 bits of IXH, IXM and IXL (index register) in the system register and store the result in IXH, IXM and IXL.

```
  IXL ← IXL + 1
  IXM ← IXM + CY
  IXH ← IXH + CY
INC IX
```

This operation can also be performed using add instructions, as follows:

```
ADD  IXL, #01H
ADDC IXM, #00H
ADDC IXH, #00H
```

Example 2

To zeroize the entire contents of data memory addresses 0.00H to 0.73H using the index register.

```
            MOV     BANK, #00H   ; Data memory bank set to 0
            MOV     IXH,  #00H   ; Set index register contents to 00H of bank 0.
            MOV     IXM, #00H    ;
            MOV     IXL,  #00H
RAM clear:
MEM000      MEM     0.00H
            SET1    IXE          ; IXE flag ← 1
            MOV     MEM000, #00H ;} Writes 0 to data memory indicated
                                 ;} by the index register
            CLR1    IXE          ; IXE flag ← 0
            INC     IX
            SKNE    IXM,  #07H   ;} Loop until index register contents
            SKE     IKL,  #04H   ;} become 74H of bank 0.
            BR      RAM clear
```

5.11 SKE m, #i Skip if data memory equal to immediate data

① Operation code

01001	m_H	m_L	i

② Function

(M) - i, skip if zero

Skips the next instruction if the contents of the data memory addressed by M are equal to the immediate data i.

③ Example

If the address 0.24H contents are 0, 0FH is transferred to address 0.24H and if not 0, it is branched to OPE1.

```
MEM024  MEM  0.24H
        MOV  BANK, #00H     ; Data memory bank set to 0
        SKE  MEM024, #00H
        BR   OPE1
        MOV  MEM024, #0FH
OPE1 :
```

5.12 SKGE m, #i Skip if data memory greater than or equal to immediate data

① Operation code

11001	m_H	m_L	i

② Function

(M) - i, skip if not borrow

Skips the next instruction if the contents of the data memory addressed by M are equal to or greater than the value of the immediate data i.

③ Example

If the address 0.1FH contents are greater than immediate data '7', 01H is stored at address 0.0FH and if smaller, 02H is stored.

```
MEM00F  MEM    0.0FH
MEM01F  MEM    0.1FH
        MOV    BANK, #00H      ; Data memory bank set to 0
        MOV    MEM00F, #01H
        SKGE   MEM01F, #07H
        MOV    MEM00F, #02H
```

5.13 SKLT m, #i Skip if data memory less than immediate data

① Operation code

11011	m_H	m_L	i

② Function

(M) - i, skip if borrow

Skips the next instruction if the contents of the data memory addressed by M are less than the value of the immediate data i.

③ Example

If the 8-bit data stored at address 0.10H (high) and address 0.20H (low) is greater than '16H', it is RETed and if smaller, it is RETSKed.

```
MEM010  MEM   0.10H
MEM020  MEM   0.20H
        MOV   BANK, #00H          ; Data memory bank set to 0
        SKGE  MEM010, #01H
        RETSK
        SKNE  MEM010, #01H
        SKLT  MEM020, #06H+01H
        RET
        RETSK
```

The same contents as above are performed as follows using the compare flag (CMP) and carry flag (CY).

```
MEM010  MEM    0.10H
MEM020  MEM    0.20H
        MOV    BANK, #00H        ; Data memory bank set to 0
        SET1   CMP
        SUB    MEM020, #06H+01H
        SUBC   MEM010, #01H
        SKT1   CY
        RET
        RETSK
```

5.14 SKNE m, #i   Skip if data memory not equal to immediate data

① Operation code

01011	m_H	m_L	i

② Function

(M) - i, skip if not zero

Skips the next instruction if the contents of the data memory addressed by M are not equal to the immediate data i.

③ Example

This program jumps to XYZ if the contents of address 0.1FH are 1 and the contents of address 0.1EH are 3, or otherwise jumps to ABC.

For an 8-bit comparison, the following combination is used.

	3		1
0.1EH	0 0 1 1	0.1FH	0 0 0 1

```
MEM01E  MEM   0.1EH
MEM01F  MEM   0.1FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        SKNE  MEM01F, #01H
        SKE   MEM01E, #03H
        BR    ABC
        BR    XYZ
```

The above processing can be coded as shown below using the compare flag and zero flag.

```
MEM01E  MEM   0.1EH
MEM01F  MEM   0.1FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        SET2  CMP, Z           ; CMP flag ← 1, Z flag ← 1
        SUB   MEM01F, #01H
        SUBC  MEM01E, #03H
        SKT1  Z
        BR    ABC
        BR    XYZ
```

5.15 AND m, #i AND between data memory and immediate data

① Operation code

10100	m_H	m_L	i

② Function

M ← (M) AND i

Finds the logical product (AND) of the contents of the data memory addressed by M and the immediate data i, and stores the result in the data memory addressed by M.

③ Example 1

To reset bit 3 (MSB) of address 0.03H.

0.03H ← (0.03H) AND 0111B

Address 0.03H

0	×	×	×

× : don't care

```
MEM003  MEM   0.03H
        MOV   BANK, #00H      ; Data memory bank set to 0
        AND   MEM003, #0111B
```

Example 2

All the address 0.03H bits should be reset for the following two instructions.

```
MEM003  MEM  0.03H
        MOV  BANK, #00H        ; Data memory bank set to 0
        AND  MEM003, #0000B
     or
MEM003  MEM  0.03H
        MOV  BANK, #00H        ; Data memory bank set to 0
        MOV  MEM003, #00H
```

5.16 AND r, m AND between general register and data memory

① Operation code

00100	m_H	m_L	r

② Function

R ← (R) AND (M)

Finds the logical product (AND) of the contents of the general register indicated by R and the contents of the data memory addressed by M, and stores the result in the general register indicated by R.

③ Example 1

To AND the contents (1010B) of address 0.03H and the contents (0110B) of address 0.2FH, and store the result (0010B) in address 0.03H.

0.03H ← (0.03H) AND (0.2FH)

```
| 1 | 0 | 1 | 0 |    Address 03H
       AND
| 0 | 1 | 1 | 0 |    Address 2FH
        ↓
| 0 | 0 | 1 | 0 |    Address 03H
```

```
MEM003  MEM  0.03H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H        ; Data memory bank set to 0
        MOV  MEM003, #1010B
        MOV  MEM02F, #0110B
        AND  MEM003, MEM02F
```

5.17 OR m, #i OR between data memory and immediate data

① Operation code

10110	m_H	m_L	i

② Function

M ← (M) OR i

Finds the logical sum (OR) of the contents of the data memory addressed by M and immediate data i, and stores the result in the data memory addressed by M.

③ Example 1

To set bit 3 (MSB) of address 0.03H.

0.03H ← (0.03H) OR 1000B

Address 0.03H

1	×	×	×

× × : don't care

```
MEM003  MEM  0.03H
        MOV  BANK, #00H      ; Data memory bank set to 0
        OR   MEM003, #1000B
```

Example 2

All the address 0.03H bits should be reset for the following two instructions.

```
MEM003  MEM  0.03H
        MOV  BANK, #00H      ; Data memory bank set to 0
        OR   MEM003, #1111B
     or
MEM003  MEM  0.03H
        MOV  BANK, #00H      ; Data memory bank set to 0
        MOV  MEM003, #0FH
```

5.18 OR r, m OR between general register and data memory

① Operation code

00110	m_H	m_L	r

② Function

R ← (R) OR (M)

Finds the logical sum (OR) of the contents of the general register indicated by R and the contents of the data memory addressed by M, and stores the result in the general register indicated by R.

③ Example 1

To OR the contents (1010B) of address 0.03H and the contents (0111B) of address 0.2FH, and store the result (1111B) in address 0.03H.

0.03H ← (0.03H) OR (0.2FH)

1	0	1	0	Address 03H
	OR			
0	1	1	1	Address 2FH
	↓			
1	1	1	1	Address 03H

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        MOV   MEM003, #1010B
        MOV   MEM02F, #0111B
        OR    MEM003, MEM02F
```

5.19 XOR m, #i Exclusive OR between data memory and immediate data

① Operation code

10101	m_H	m_L	i

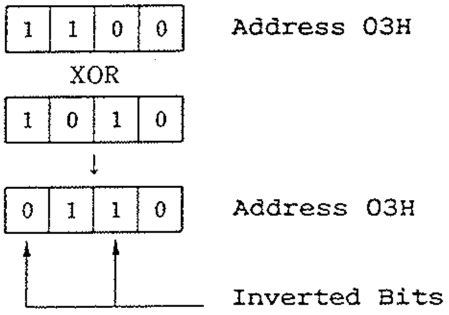
② Function

M ← (M) XOR i

Finds the exclusive logical sum (XOR) of the contents of the data memory addressed by M and the immediate data i, and stores the result in the data memory addressed by M.

③ Example

To invert bit 1 and bit 3 of address 0.03H and store the result in address 03H.



```
MEM003  MEM  0.03H
        MOV  BANK, #00H      ; Data memory bank set to 0
        XOR  MEM003, #1010B
```

5.20 XOR r, m Exclusive OR between general register and data memory

① Operation code

00101	m_H	m_L	r

② Function

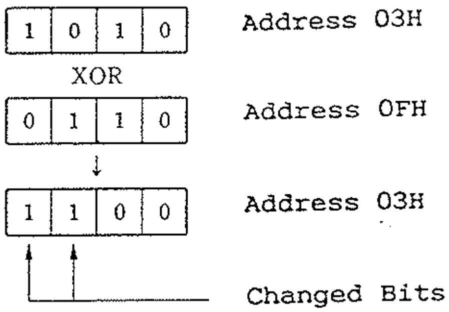
$R \leftarrow (R)$ XOR (M)

Finds the exclusive logical sum (XOR) of the contents of the general register indicated by R and the contents of the data memory addressed by M, and stores the result in the general register indicated by R.

③ Example 1

This program compares the contents of address 0.03H with the contents of address 0.0FH, sets differing bits, and stores the result in address 0.03H; if all bits of address 0.03H are reset (the contents of address 0.03H and address 0.0FH are the same) the program jumps to LBL1, otherwise it jumps to LBL2.

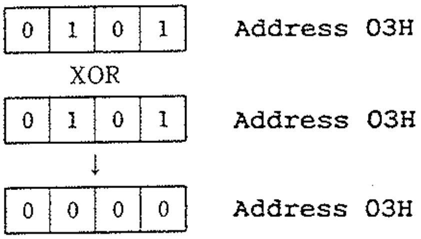
This example can be used, for instance, to compare the alternate switch status (contents of address 0.03H) with the internal status (contents of address 0.0FH), and jump to processing for a switch which has changed.



```
MEM003  MEM   0.03H
MEM00F  MEM   0.0FH
        MOV   BANK, #00H        ; Data memory bank set to 0
        XOR   MEM003, MEM00F
        SKNE  MEM003, #00H
        BR    LBL1
        BR    LBL2
```

Example 2

To clear the contents of address 0.03H.



```
MEM003  MEM   0.03H
        MOV   BANK, #00H
        XOR   MEM003, MEM003
```

5.21 LD r, m Load data memory to general register

1 Operation code

01000	m_H	m_L	r

2 Function

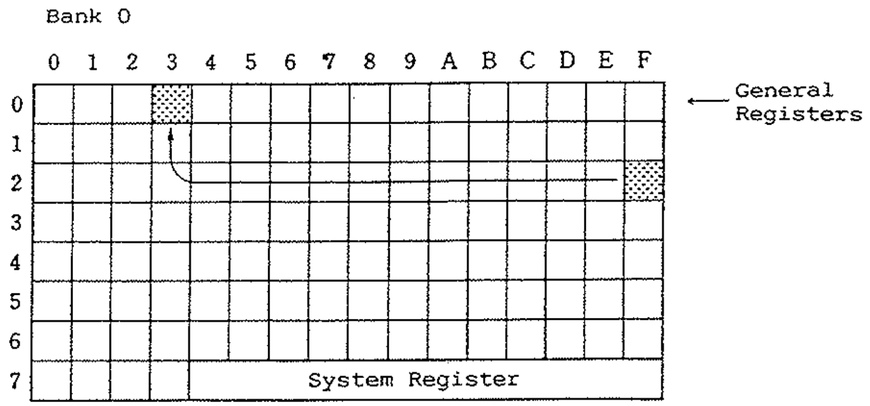
R ← (M)

Stores the contents of the data memory addressed by M in the general register indicated by R.

3 Example 1

To store the contents of address 0.2FH in address 0.03H.

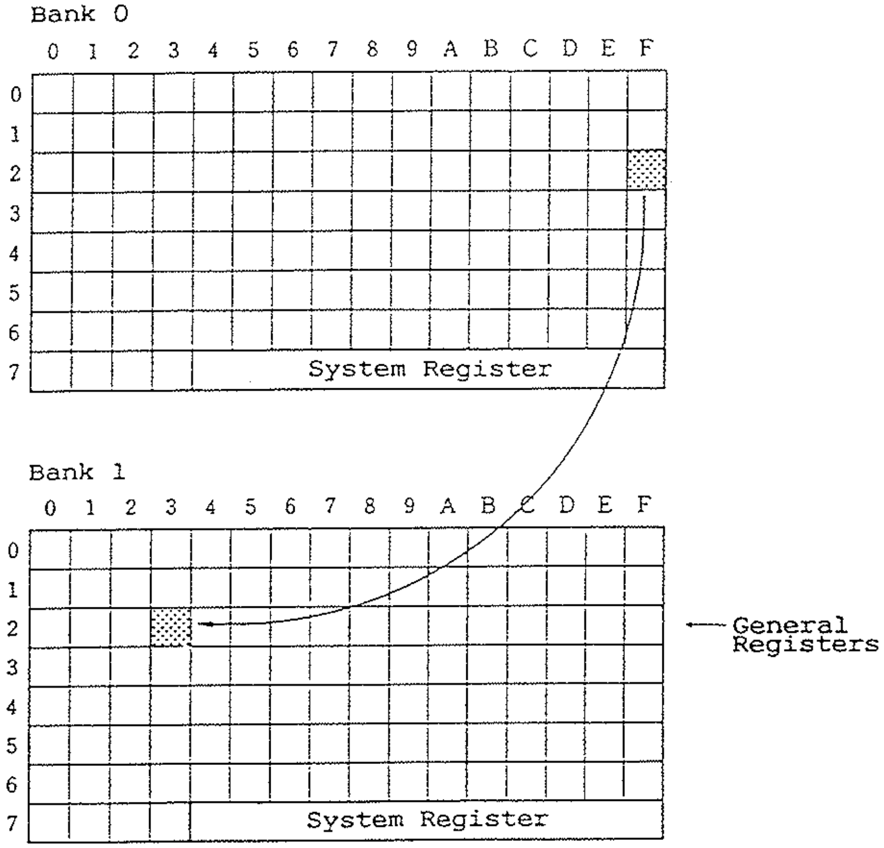
```
        0.03H ← (0.2FH)
MEM003  MEM  0.03H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H     ; Data memory bank set to 0
        LD   MEM003, MEM02F
```



Example 2

To store the contents of address 0.2FH in address 1.23H when bank 1 row address 2 (1.20H to 1.2FH) is specified as general registers (RPH = 1, RPL = 4).

```
        1.23H ← (0.2FH)
MEM123  MEM   1.23H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H      ; Data memory bank set to 0
        MOV   RPH, #01H       ; General register bank set to 1
        MOV   RPL, #04H       ; General register row address set to 2
        LD    MEM123, MEM02F
```



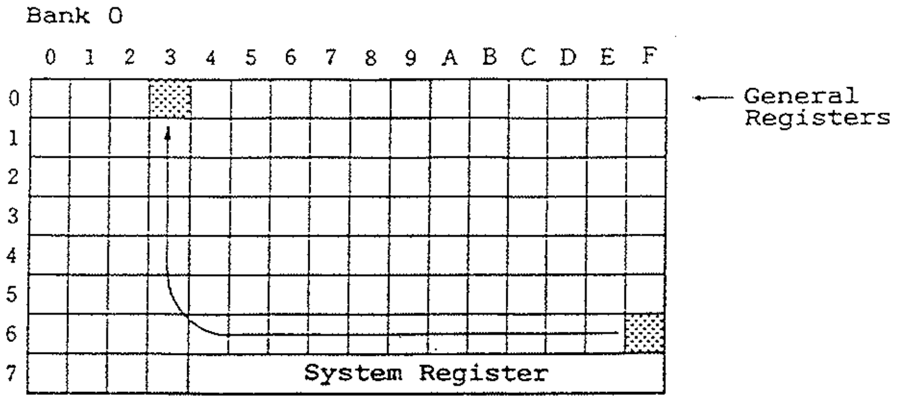
Example 3

To store the contents of address 0.6FH in address 0.03H. If IXE = 1, IXH = 0, IXM = 4 and IXL = 0, that is IX = 0.40H, data memory 0.6FH can be specified by making the data memory address 2FH.

```
IXH ← 00H
IXM ← 04H
IXL ← 00H
IXE flag ← 1
0.03H ← (0.6FH)
```

Address obtained by ORing index register contents 0.40H and data memory 0.2FH

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        MOV   IXH, #00H        ; IX ← 00001000000B (0.40H)
        MOV   IXM, #04H
        MOV   IXL, #00H
        SET1  IXE              ; IXE flag ← 1
        LD    MEM003, MEM02F
```



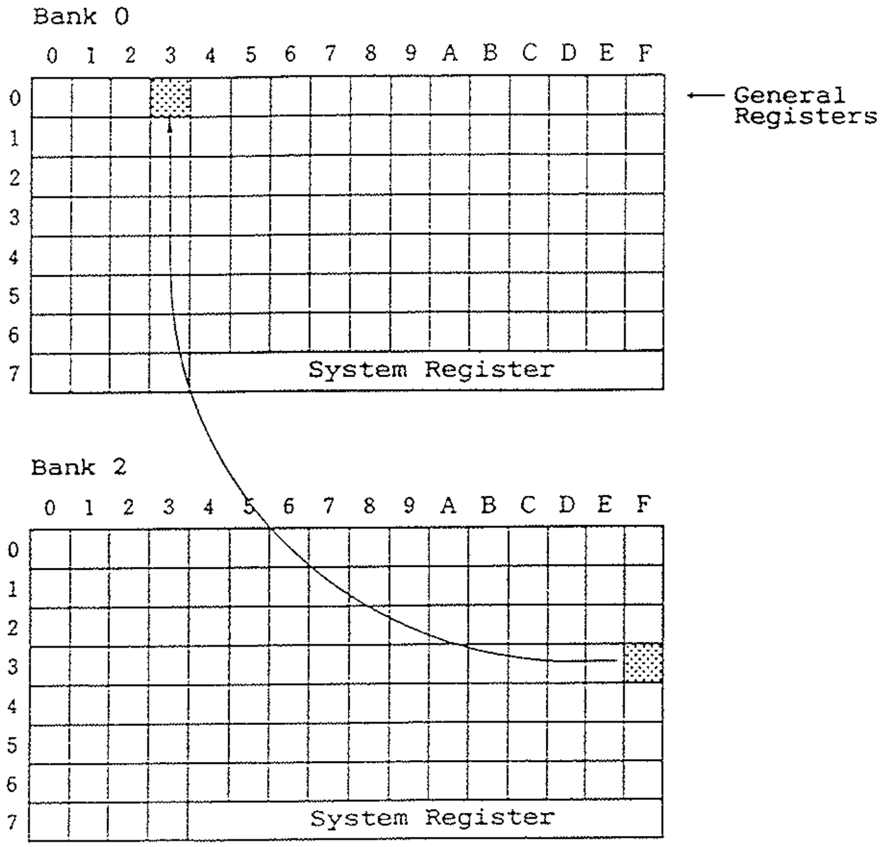
Example 4

To store the contents of address 2.3FH in address 0.03H. If IXE = 1, IXH = 1, IXM = 1 and IXL = 0, that is IX = 2.10H, data memory 2.3FH can be specified by making the data memory address 2FH.

0.03H ← (2.3FH)

Address obtained by ORing index register contents 2.10H and data memory 0.2FH

```
MEM003  MEM   0.03H
MEM02F  MEM   0.2FH
        MOV   BANK, #00H        ; Data memory bank set to 0
        MOV   IXH, #01H         ; IX ← 00100010000B (2.10H)
        MOV   IXM, #01H
        MOV   IXL, #00H
        SET1  IXE               ; IXE flag ← 1
        LD    MEM003, MEM02F
```



④ Note

The first operand of the "LD r, m" instruction is the general register column address. Thus, when the following coding is used, the general register column address is 03H.

```
MEM013  MEM  0.13H
MEM02F  MEM  0.2FH
        LD   MEM013, MEM02F
             ~~~~~~
               |
               └──────── Indicates the general register column
                         address:  The low-order 4 bits are
                         valid.  In this case, if bank 0
                         row address 0 is specified as the
                         general registers, address 03H is
                         specified.
```

5.22 ST m, r Store general register to data memory

① Operation code

11000	m_H	m_L	r

② Function

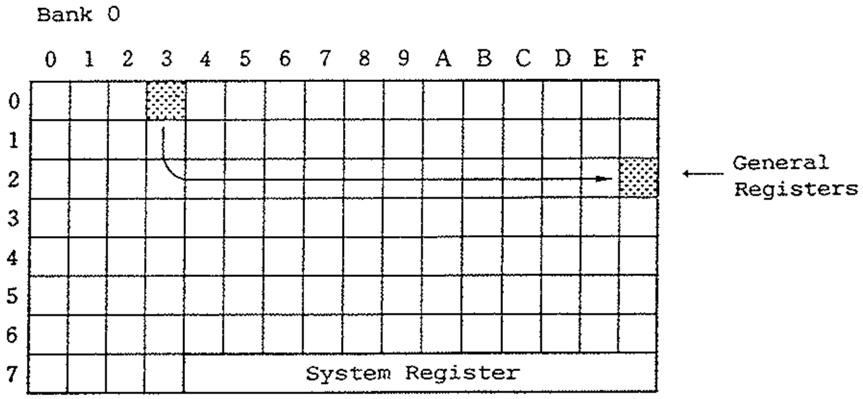
M ← (R)

Stores the contents of the general register indicated by R in the data memory addressed by M.

③ Example 1

To store the contents of address 0.03H in address 0.2FH.

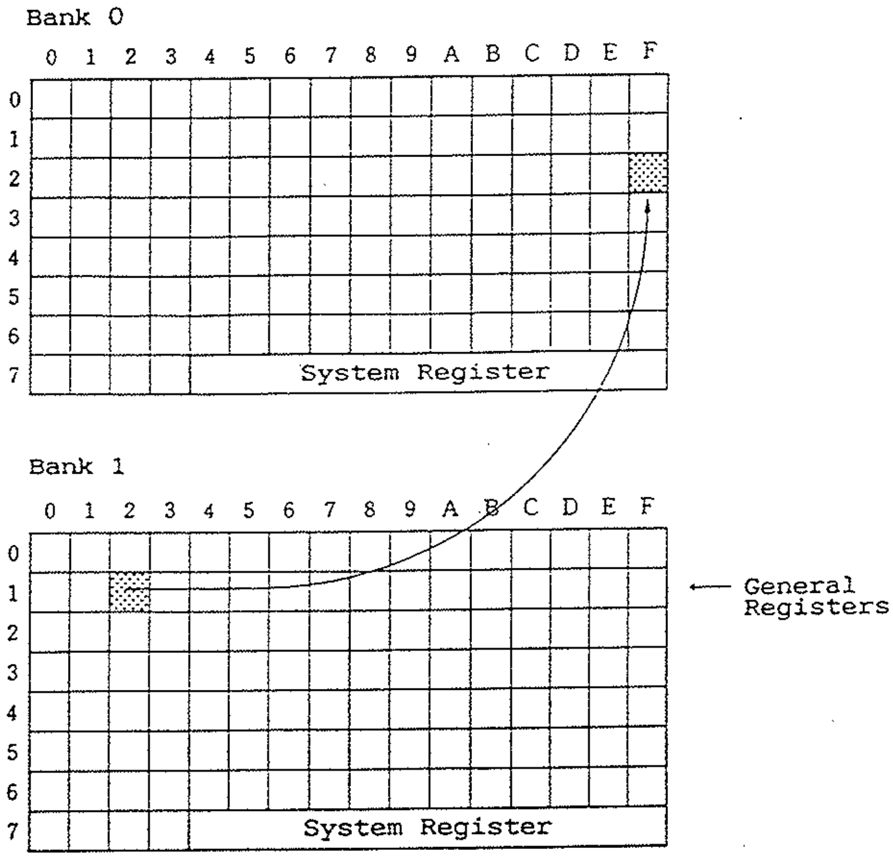
```
MEM003  MEM  0.03H
MEM02F  MEM  0.2FH
        MOV  BANK, #00H       ; Data memory bank set to 0
        ST   MEM02F, MEM003 ; Transfer general register contents
                              to data memory.
```



Example 2

To store the contents of address 1.13H in address 0.2FH. The general registers are specified as bank 1 row address 1 (1.10H to 1.1FH) by the register pointer.

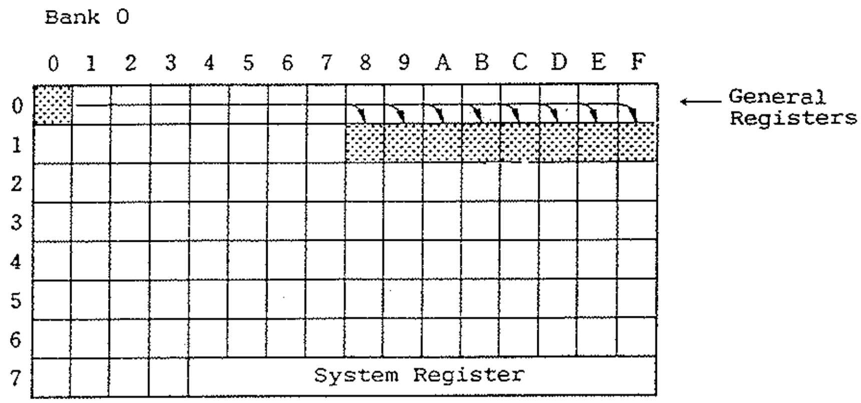
```
        (0.2FH) ← (1.13H)
MEM02F  MEM  0.2FH
MEM113  MEM  1.13H
        MOV  BANK, #00H       ; Data memory bank set to 0
        MOV  RPH, #01H        ; General register bank set to 1
        MOV  RPL, #02H        ; General register row address set to 1
        ST   MEM02F, MEM113 ; Transfer general register contents to
                              data memory.
```



Example 3

To store the contents of address 0.00H in addresses 0.18H through 0.1FH. The data memory (18H to 1FH) is specified by the index register.

```
 (0.18H)  ← (0.00H)
 (0.19H)  ← (0.00H)
             :
             :
             :
 (0.1FH)  ← (0.00H)
          MOV    BANK, #00H        ; Data memory bank set to 0
          MOV    IXH, #00H         ; IX ← 00000000000B (0.00H)
          MOV    IXM, #00H
          MOV    IXL, #00H         ; Address 0.00H specified as
                                     data memory
MEM018    MEM    0.18H
MEM000    MEM    0.00H
LOOP1 :
          SET1   IXE               ; IXE flag ← 1
          ST     MEM018, MEM000 ;  (0.1×H) ← (0.00H)
          CLR1   IXE               ; IXE flag ← 0
          INC    IX                ; Index register + 1
          SKGE   IXL, #08H
          BR     LOOP1
```



5.23 MOV @r, m Move data memory to destination indirect

① Operation code

01010	m_H	m_L	r

② Function

When MPE = 1

$[(MP), (R)] \leftarrow (M)$

When MPE = 0

$[m_H, (R)] \leftarrow (M)$

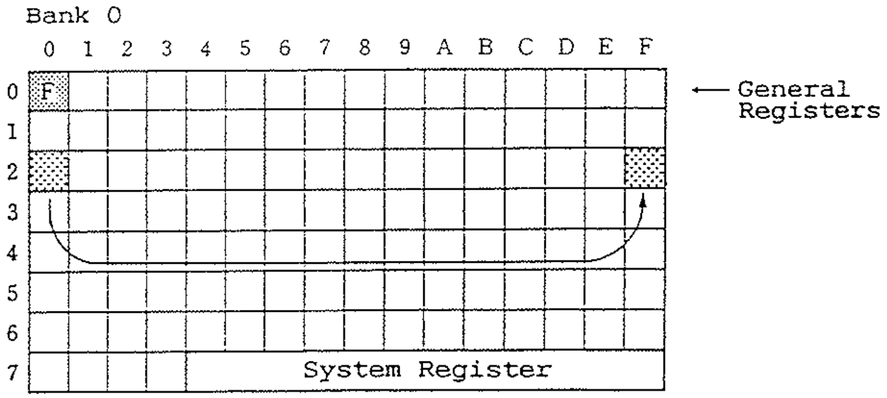
Stores the contents of the data memory addressed by M in the data memory indicated by the general register R.

When MPE = 0, the transfer is performed between locations with the same row address in the same bank.

③ Example 1

To store the contents of address 0.20H in address 0.2FH. The storage destination data memory is specified by the column address indicated by the general register (00H) and the data memory (20H) row address.

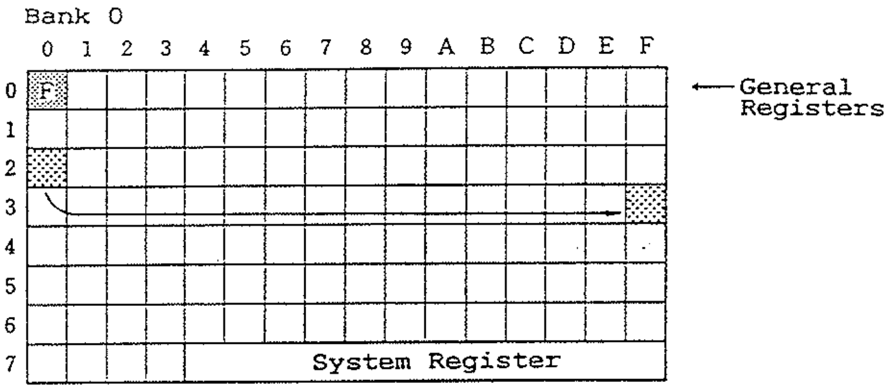
```
                (0.2FH) ← (0.20H)
MEM000  MEM  0.00H
MEM020  MEM  0.20H
        MOV  BANK, #00H        ; Data memory bank set to 0
        CLR1 MPE               ; MPE flag ← 0
        MOV  MEM000, #0FH      ; Column address set in general
                                 register
        MOV  @MEM000, MEM020 ; Store contents.
```



Example 2

To store the contents of address 0.20H in address 0.3FH. The storage destination data memory is specified by the column address indicated by the general register (00H) and the row address indicated by the memory pointer (MP).

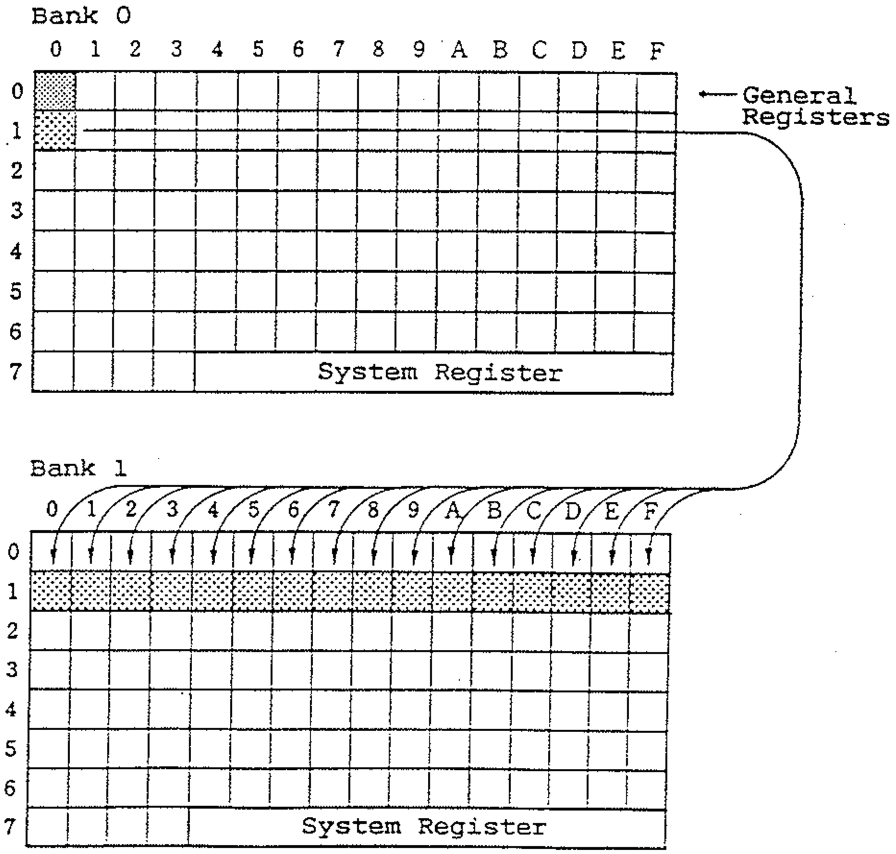
```
          (0.3FH) ← (0.20H)
MEM000  MEM    0.00H
MEM020  MEM    0.20H
        MOV    BANK, #00H        ; Data memory bank set to 0
        MOV    RPH, #00H         ; General register bank set to 0
        MOV    RPL, #00H         ; General register row address set
                                   to 0
        MOV    MEM000, #0FH      ; Column address set in general
                                   register
        MOV    MPH, #08H         ;⎫ Set bank 0 and row address 3 in
        MOV    MPL, #03H         ;⎭ memory pointer.
        ;SET1  MPE               ; MPE flag (bit 3 of MPH) ← 1
        MOV    @MEM000, MEM020 ; Store contents.
```



Example 3

To store the contents of address 0.10H in addresses 1.10H through 1.1FH.

```
(1.10H) ← (0.10H)
(1.11H) ← (0.10H)
          :
          :
          :
(1.1FH) ← (0.10H)
MEM000  MEM    0.00H
MEM010  MEM    0.10H
        MOV    BANK, #00H         ; Data memory bank set to 0
        MOV    RPH, #00H          ; General register bank set to 0
        MOV    RPL, #00H          ; General register row address set
                                    to 0
        MOV    MEM000, #00H       ; Column address set in general
                                    register
        MOV    MPH, #08H          ;⎫ Set bank 1 and row address 1 in
        MOV    MPL, #09H          ;⎭ memory pointer.
        ; SET1 MPE                ; MPE flag (bit 3 of MPH) ← 1
LOOP1 :
        MOV    @MEM000, MEM010 ; [(MP), (00H)] ← (10H)
        ADD    MEM000, #01H       ; Column address + 1
        SKT1   CY                 ; Finished up to address 1FH in
                                    bank 1?
        BR     LOOP1
```


Bank 0
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7
General Registers
System Register
Bank 1
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7
System Register

5.24 MOV m, @r Move data memory to destination indirect

① Operation code

11010	m_H	m_L	r

② Function

When MPE = 1

$(M) \leftarrow [(MP), (R)]$

When MPE = 0

$(M) \leftarrow [m_H, (R)]$

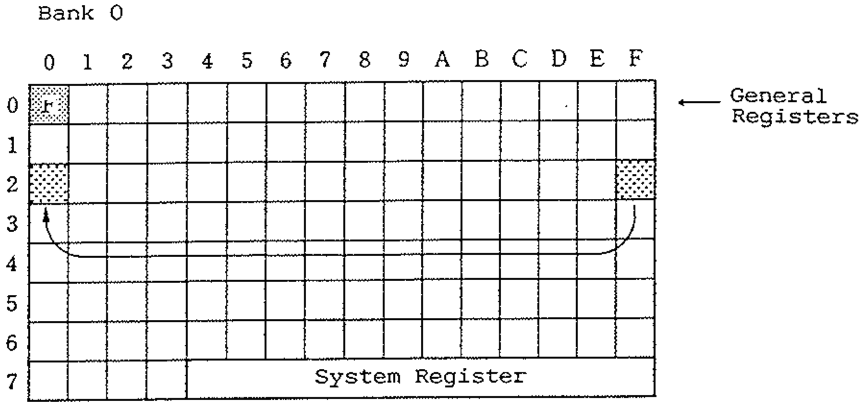
Stores the contents of the data memory indicated by the general register R in the data memory addressed by M.

When MPE = 0, the transfer is performed between locations with the same row address in the same bank.

③ Example 1

To store the contents of address 0.2FH in address 0.20H. The storage destination data memory is specified by the column address indicated by the general register (00H) and the data memory (20H) row address.

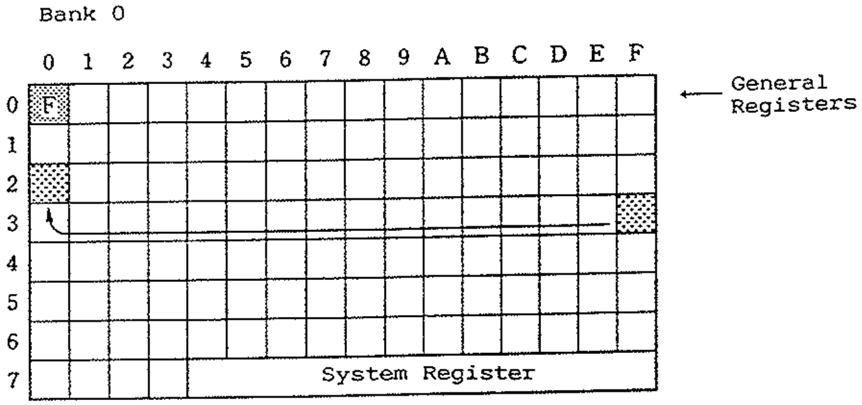
```
                (0.20H) ← (0.2FH)
MEM000   MEM   0.00H
MEM020   MEM   0.20H
         MOV   BANK, #00H          ; Data memory bank set to 0
         CLR1  MPE                 ; MPE flag ← 0
         MOV   MEM000, #0FH        ; Column address set in general
                                     register
         MOV   MEM020, @MEM000 ;   Store contents.
```



Example 2

To store the contents of address 0.3FH in address 0.20H. The storage destination data memory is specified by the column address indicated by the general register (00H) and the row address indicated by the memory pointer (MP).

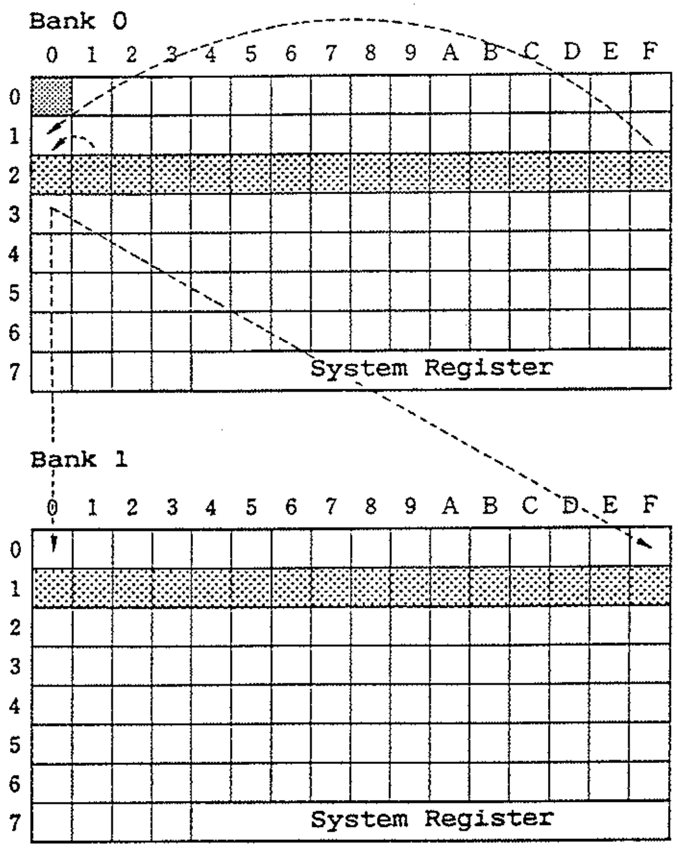
```
                (0.20H) ← (0.3FH)
MEM000  MEM     0.00H
MEM020  MEM     0.20H
        MOV     BANK, #00H          ; Data memory bank set to 0
        MOV     MEM000, #0FH        ; Column address set in general
                                      register
        MOV     MPH, #08H           ;} Set bank 0 and row address 3 in
        MOV     MPL, #03H           ;} memory pointer.
        ; SET1  MPE                 ; MPE flag (bit 3 of MPH) ← 1
        MOV     MEM020, @MEM000 ; Store contents.
```



Example 3

To store the contents of address 0.20H through 0.2FH in addresses 1.10H through 1.1FH. The storage data memory is specified by the column address indicated by the general register (00H) and the memory pointer (MP) or data memory (20H) row address.

```
  (1.10H) ← (0.20H)
  (1.11H) ← (0.21H)
  (1.12H) ← (0.22H)
             ⋮
  (1.1FH) ← (0.2FH)
MEM000  MEM    0.00H
MEM020  MEM    0.20H
        MOV    BANK, #00H          ; Data memory bank set to 0
        MOV    MEM000, #00H        ; Column address set in general
                                     register
        MOV    MPH, #00H           ;⎫ Set bank 1 and row address 1 in
        MOV    MPL, #09H           ;⎭ memory pointer.
        ; CLR1 MPE                 ; MPE flag (bit 3 of MPH) ← 0
LOOP1 :
        MOV    MEM020, @MEM000 ; (20H) ← [2, (00H)]
        SET1   MPE                 ; MPE flag ← 1
        MOV    @MEM000, MEM020 ; [(MP), (00H)] ← (20H)
        CLR1   MPE                 ; MPE flag ← 0
        ADD    MEM000, #01H        ; Column address + 1
        SKT1   CY                  ; Finished up to 1FH in bank 1
        BR     LOOP1
```


Bank 0
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7
System Register
Bank 1
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7
System Register

General Registers
In this case, data is
transferred via 20H.

5.25 MOV m, #i Move immediate data to data memory

① Operation code

11101	m_H	m_L	i

② Function

(M) ← i

Stores the immediate data i in the data memory addressed by M.

③ Example 1

To store the immediate data OAH in address 0.50H specified as data memory.

```
                0.50H ←0AH
  MEM050   MEM   0.50H
           MOV   BANK, #00H      ; Data memory bank set to 0
           MOV   MEM050, #0AH
```

Example 2

Immediate data 07H is stored as address 0.32H contents. At this time, if IXE = 1, IXH = 0, IXM = 3, IXL = 2, that is, IX = 0.32H, data memory 0.32H can be specified by setting the data memory address to 0.00H.

```
0.32H ←07H
  └─ Address determined by index register contents 0.32H
     ORed with data memory address 0.00H.
```

```
MEM000  MEM   0.00H
        MOV   IXH, #00H        ; IX ← 00000110010B (0.32H)
        MOV   BANK, #00H       ; Data memory bank set to 0
        MOV   IXM, #03H
        MOV   IXL, #02H
        SET1  IXE              ; IXE flag ← 1
        MOV   MEM000, #07H
```

5.26 MOVT DBF, @AR Move program memory data specified by AR to DBF

① Operation code

00111	000	0001	0000

② Function

```
SP ← (SP) - 1,
ASR ← PC,
PC ← (AR),
DBF ← (AR) rom,
PC ← (ASR),
SP ← (SP) + 1
```

Stores the contents of the program memory addressed by the address register AR in the data buffer DBF.

Since this instruction temporarily uses one stack level, care is required concerning nesting of subroutines, interrupts, etc.

③ Example 1

To transfer 16-bit table data determined by the value of the address register (AR3, AR2, AR1, AR0) in the system register to the data buffer (DBF3, DBF2, DBF1, DBF0).

```
            ; *
            ; * * Table data
            ; *
Address     ORG    0010H
 0010H      DW     0000000000000000B ; (0000H)
 0011H      DW     1010101111001101B ; (0ABCDH)
                          :
                          :
                          :
            ; *
            ; * * Table reference program
            ; *
            MOV    AR3, #00H      ; AR3 ← 00H   Sets 0011H in address
                                                register
            MOV    AR2, #00H      ; AR2 ← 00H
            MOV    AR1, #01H      ; AR1 ← 01H
            MOV    AR0, #01H      ; AR0 ← 01H
            MOVT   DBF, @AR       ; Transfers data in address 0011H
                                    to DBF
```

In this case, the data shown below is stored in the DBF.

DBF3 = 0AH

DBF2 = 0BH

DBF1 = 0CH

DBF0 = 0DH

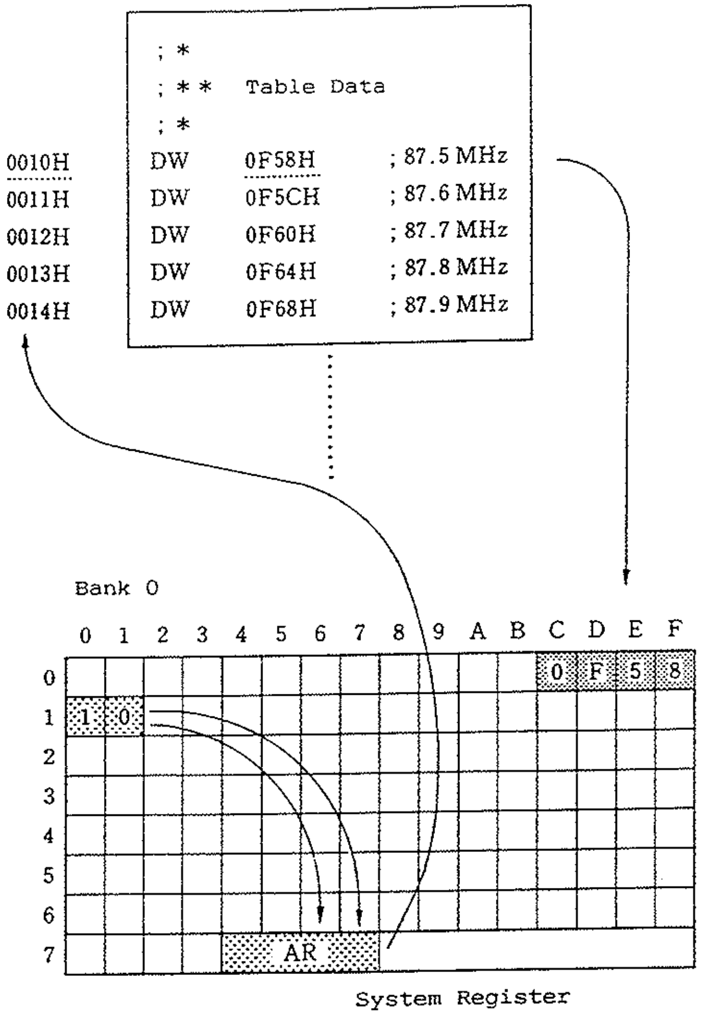
Example 2

Set the channel number as data memory to addresses 0.10H and 0.11H. According to those contents, obtain the PLL frequency division value (N value) and transfer it to the PLL data register (PLLR). However, it is presumed that the intermediate frequency is 10.7 MHz and 25 kHz is selected as a reference frequency.

```
          ; *
          ; * *  N value table data
          ; *
Address   ORG      0010H
0010H     DW       0F58H          ; 87.5 MHz (Lowest frequency  Channel 00)
0011H     DW       0F5CH          ; 87.6 MHz
0012H     DW       0F60H          ; 87.7 MHz
0013H     DW       0F64H          ; 87.8 MHz
0014H     DW       0F68H          ; 87.9 MHz
0015H     DW       0F6CH          ; 88.0 MHz
0016H     DW       0F70H          ; 88.1 MHz
0017H     DW       0F74H          ; 88.2 MHz
                     :
                     :
                     :

          ; *
          ; * *  N value setting program
          ; *
MEM010    MEM      0.10H
MEM011    MEM      0.11H
          MOV      BANK, #00H     ; Data memory bank set to 0
          MOV      RPH,  #00H     ; RPH ← 00H }  Set row address 7 (0.70H
          MOV      RPL,  #0EH     ; RPL ← 0EH }  to 0.7FH) as general
          MOV      AR3,  #00H     ; AR3 ← 0   }  registers.
          MOV      AR2,  #00H     ; AR2 ← 0
```

```
LD      AR1, MEM010   ; AR1 ← 10H   High-order channel data
LD      AR0, MEM011   ; AR0 ← 11H   Low-order channel data.
ADD     AR1, #01H     ;⎫
ADDC    AR2, #00H     ;⎬ As table data start address is
ADDC    AR3, #00H     ;⎭ 0010H, 0010H is added to address
                         register
MOVT    DBF, @AR      : Store table data in DBF.
PUT     PLLR, DBF     ; Transfer N value to PLL data
                        register (PLLR).
        :
        :
```



```
        ; *
        ; **   Table Data
        ; *
0010H   DW     0F58H      ; 87.5 MHz
0011H   DW     0F5CH      ; 87.6 MHz
0012H   DW     0F60H      ; 87.7 MHz
0013H   DW     0F64H      ; 87.8 MHz
0014H   DW     0F68H      ; 87.9 MHz
```

④ Note 1

The number of bits which can be used in the address register (AR3 to AR0) varies from product to product: Refer to the Data Sheet for the product concerned when using this register.

Note 2

When the "MOVT" instruction is executed, level 1 is used as the stack. Therefore special attention must be paid to the stack level when this instruction is used in a subroutine or interrupt service routine.

Note 3

It takes 2 machine cycles to execute one instruction only in the case of the "MOVT" instruction. Special care is required when creating a program which uses the software timer.

5.27 PUSH AR — Push address register

① Operation code

00111	000	1101	0000

② Function

$SP \leftarrow (SP) - 1,$

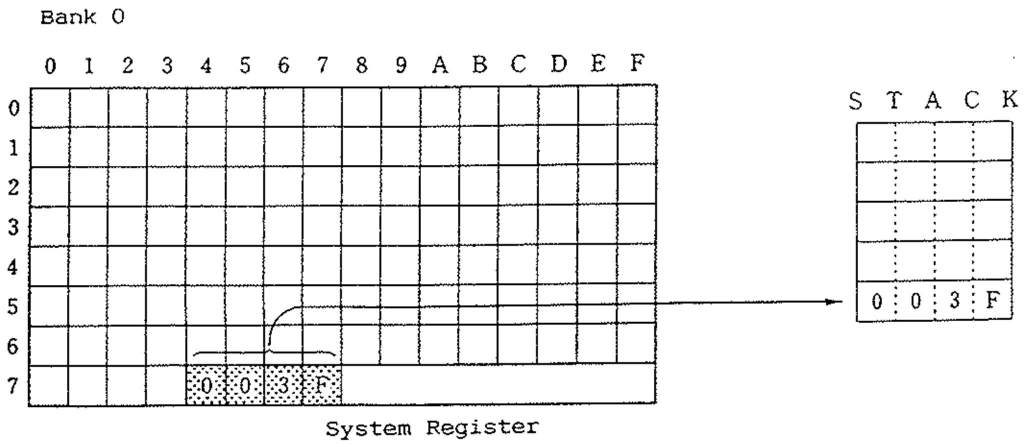
$ASR \leftarrow (AR)$

Decrements the stack pointer (SP), then stores the value of the address register (AR) in the stack.

③ Example 1

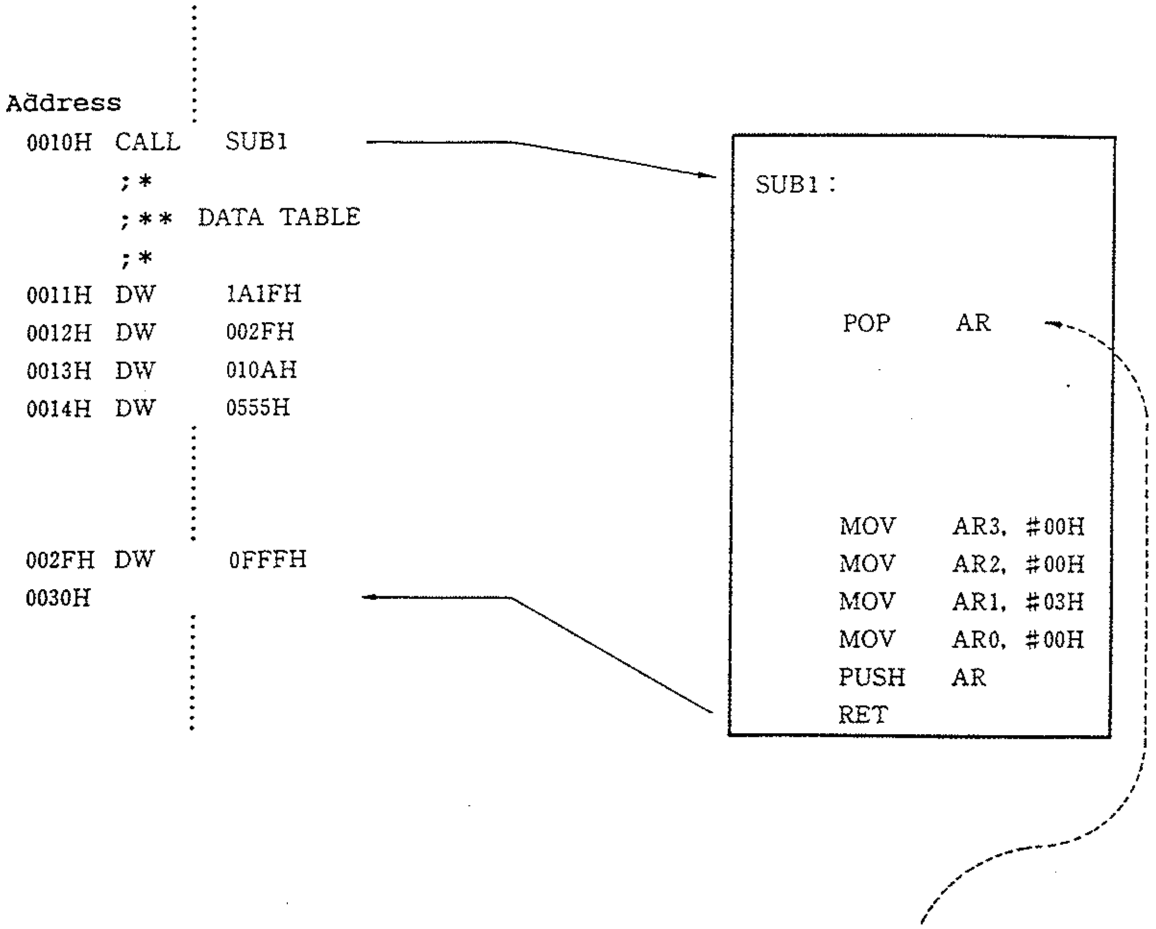
To set 003FH in the address register and store it in the stack.

```
MOV     AR3,  #00H
MOV     AR2,  #00H
MOV     AR1,  #03H
MOV     AR0,  #0FH
PUSH    AR
```



Example 2

To set the subroutine return address in the address register when there is a data table at the end of the subroutine, and return to the address.



```
Address
0010H  CALL   SUB1
       ;*
       ;**  DATA TABLE
       ;*
0011H  DW     1A1FH
0012H  DW     002FH
0013H  DW     010AH
0014H  DW     0555H

002FH  DW     0FFFH
0030H
```

```
SUB1:

          POP     AR

          MOV     AR3, #00H
          MOV     AR2, #00H
          MOV     AR1, #03H
          MOV     AR0, #00H
          PUSH    AR
          RET
```

If a "POP" instruction is executed at this point, the contents of the address register are "0011H" (address after CALL instruction).

5.28 POP AR

Pop address register

① Operation code

00111	000	1100	0000

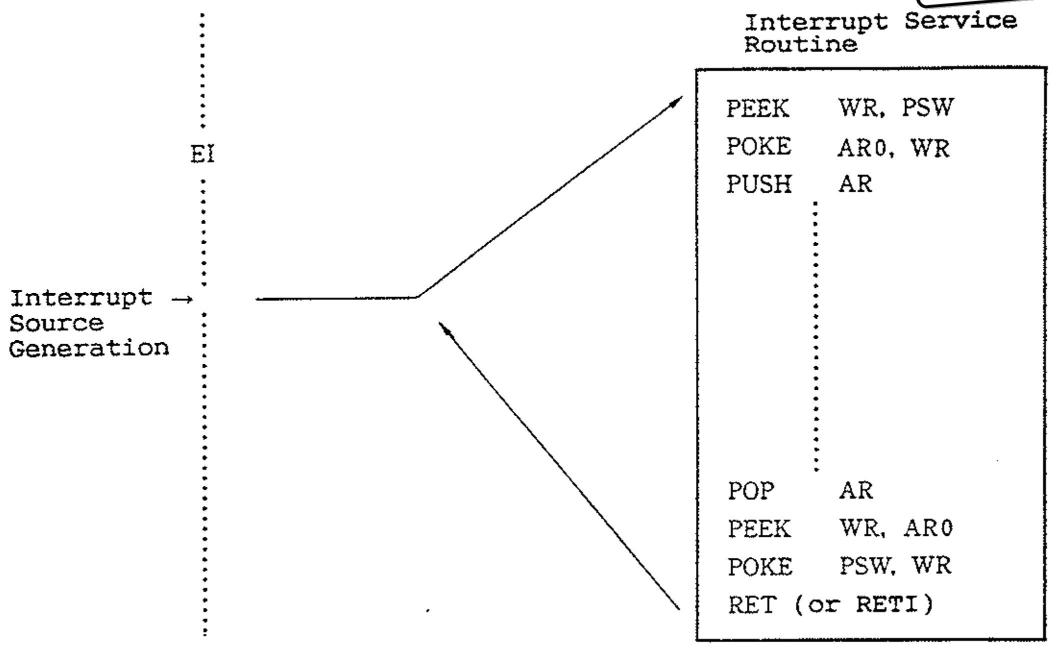
② Function

AR ← (ASR),
SP ← (SP) + 1

Fetches the stack contents into the address register (AR), then increments the stack pointer (SP).

③ Example

In this example, if the PSW is changed within the interrupt service routine when interrupt servicing is performed, at the start of the interrupt service routine the contents of the PSW are transferred to the address register via the WR, and saved to the stack by a "PUSH" instruction, and before returning are restored to the address register by a "POP" instruction and transferred to the PSW via the WR.



```
PEEK    WR, PSW
POKE    AR0, WR
PUSH    AR
   :
POP     AR
PEEK    WR, AR0
POKE    PSW, WR
RET (or RETI)
```

5.29 PEEK WR, rf Peek register file to window register

① Operation code

00111	rf_H	0011	rf_L

② Function

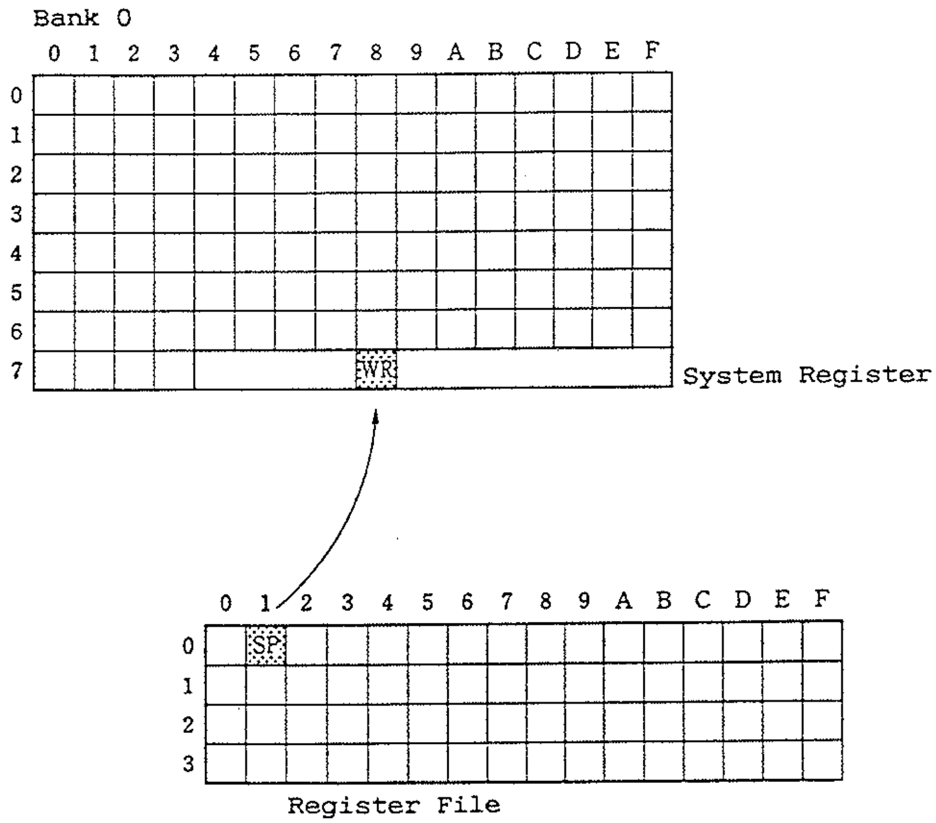
WR ← (RF)

Stores the register file contents addressed by rf in the window register WR.

③ Example 1

To store in the window register the contents of the stack pointer (SP) in address 01H in the register file.

```
PEEK   WR, SP
```


Bank 0
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7
WR
System Register
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
SP
Register File

5.30 POKE rf, WR Poke window register to register file

① Operation code

00111	rf_H	0010	rf_L

② Function

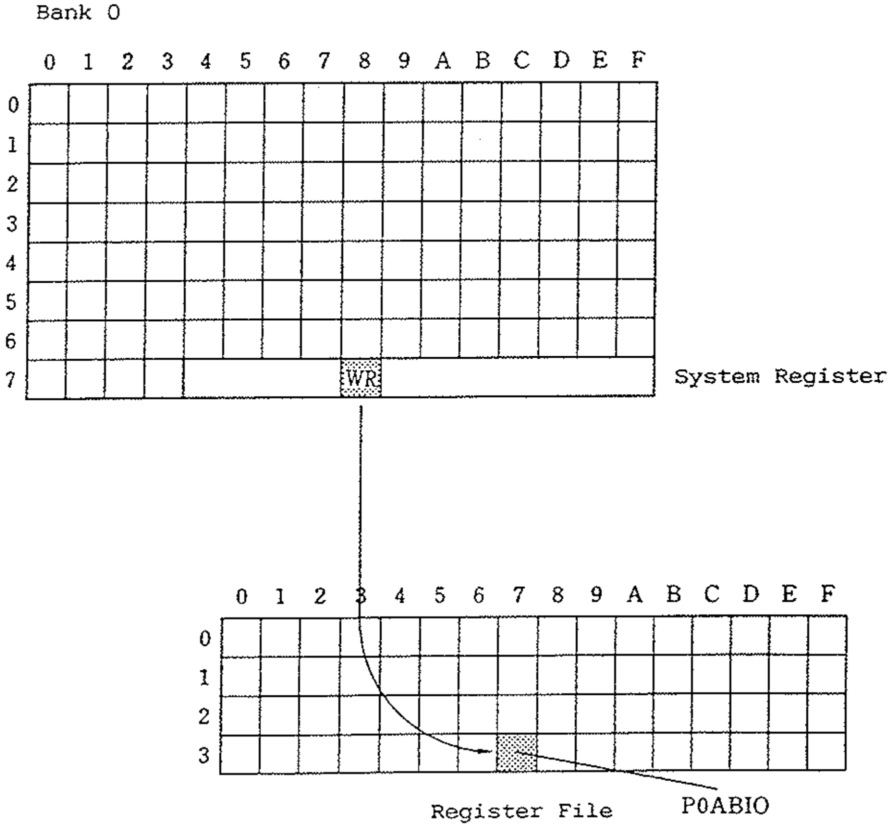
RF ← (WR)

Stores the contents of the window register WR in the register file location addressed by rf.

③ Example 1

To store immediate data 0FH in register file P0ABIO via the window register.

```
P0ABIO  MEM 0.0B7H
MOV     WR, #0FH
POKE    P0ABIO, WR    ; Set POA0, POA1, POA2, POA3 to output mode.
```

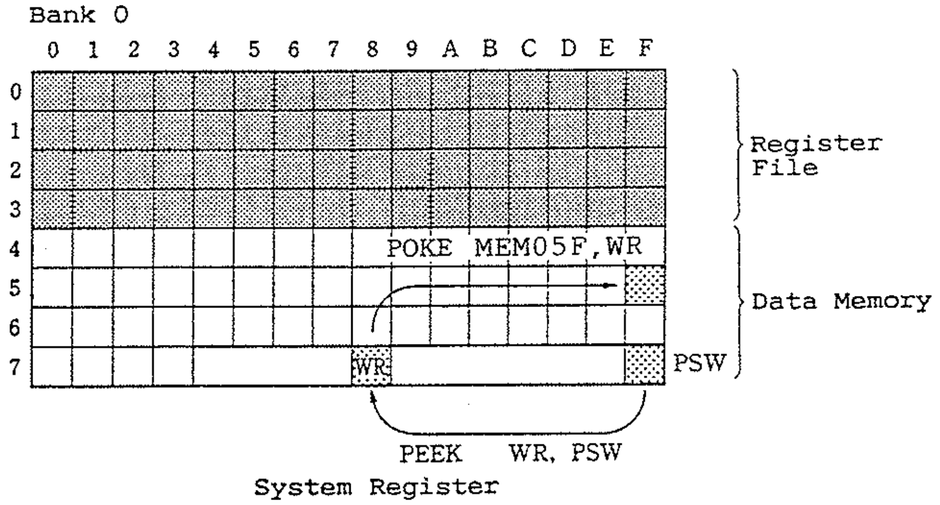


④ Note

In addition to the register file, "PEEK" and "POKE" instructions can be used to access addresses 40H through 7FH of all data memory banks. For example, these instructions may be used in the following way.

```
MEM05F  MEM   0.5FH
        PEEK  WR, PSW       ; Stores contents of PSW (7FH) in
                              system register in WR

        POKE  MEM05F, WR    ; Stores WR contents in data memory
                              address 5FH
```



5.31 GET DBF, p Get peripheral data to data buffer

① Operation code

00111	p_H	1011	p_L

② Function

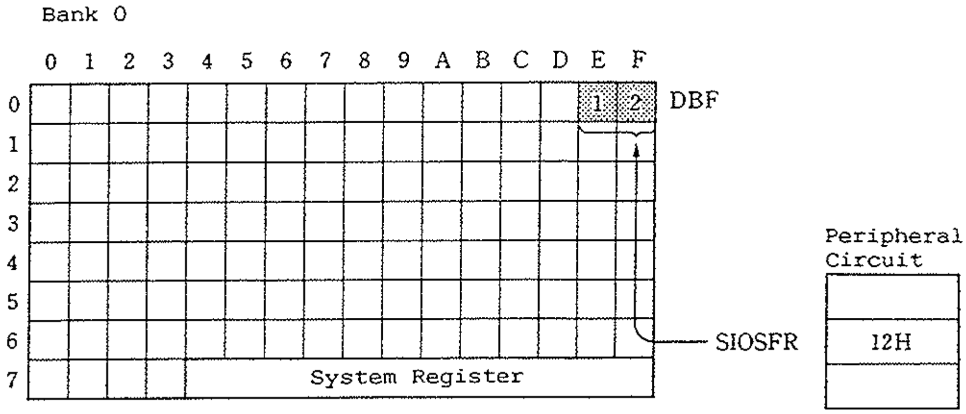
DBF ← (PE)

Stores the contents of the peripheral circuit addressed by p in the data buffer DBF.

③ Example 1

To store the contents (8 bits) of the shift register (SIOSFR) peripheral circuit in data buffer locations DBF0, DBF1.

```
GET     DBF, SIOSFR
```



④ Note 1

The data buffer is allocated to addresses OCH, ODH, OEH and OFH in data memory bank 0 irrespective of the value of the bank register.

Bank 0

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0													DBF			
1																
2																
3																
4																
5																
6																
7					System Register											

Note 2

The data buffer comprises 16 bits in total, but the number of bits used as the input/output unit differs depending on the peripheral circuit accessed by the "GET" instruction. It should be noted, therefore, that when the "GET" instruction is executed for a peripheral circuit which uses an 8-bit input/output unit, for example, the data is stored in the low-order 8 bits (DBF1, DBFO) of the data buffer DBF.

5.32 PUT p, DBF — Put data buffer to peripheral

① Operation code

00111	p_H	1010	p_L

② Function

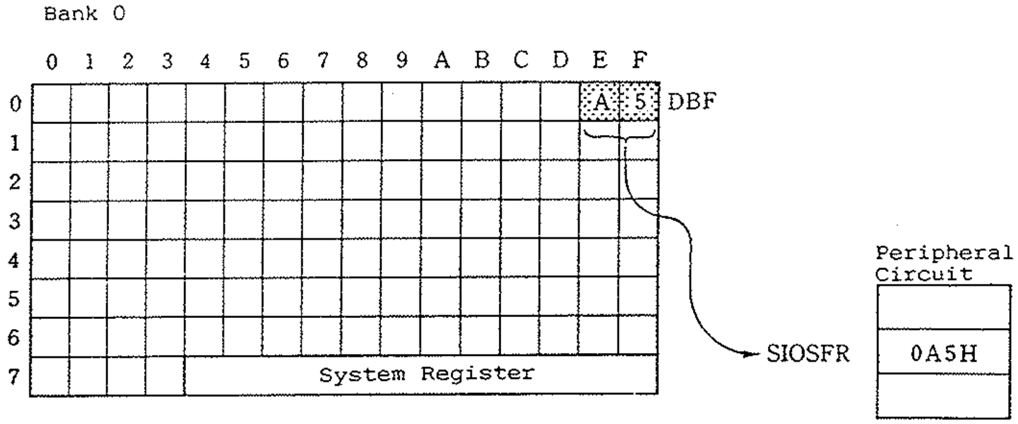
PE ← (DBF)

Stores the contents of the data buffer DBF in the peripheral circuit addressed by p.

③ Example 1

To set 0AH and 05H in data buffer locations DBF1 and DBF0 respectively, and transfer this data to the serial I/O shift register (SIOSFR) peripheral circuit.

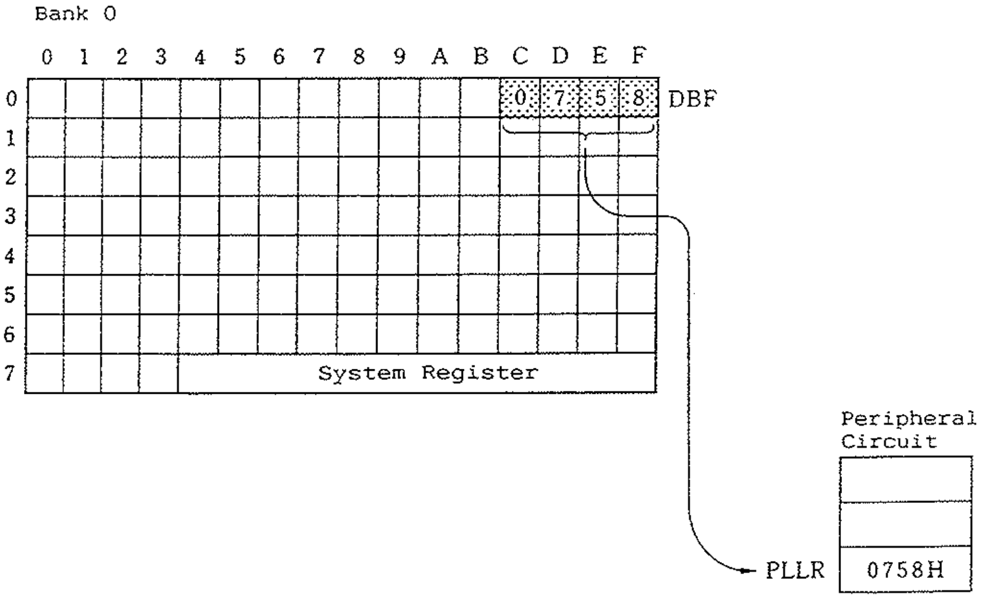
```
MOV    BANK, #00H   ; Data memory bank set to 0
MOV    DBF0,  #05H
MOV    DBF1,  #0AH
PUT    SIOSFR, DBF
```



Example 2

To set the data 0758H in data buffer locations DBF0 through DBF3 as PLL data, and transfer this data to the PLL data register (PLLR) peripheral circuit.

```
MOV     BANK, #00H  ; Data memory bank set to 0
MOV     DBF3, #00H
MOV     DBF2, #07H
MOV     DBF1, #05H
MOV     DBF0, #08H
PUT     PLLR, DBF
```



④ Note

The data buffer comprises 16 bits in total, but the number of bits used as the input/output unit differs depending on the peripheral circuit accessed by the "PUT" instruction. It should be noted, therefore, that when the "PUT" instruction is executed for the serial I/O shift register, for example, which uses an 8-bit input/output unit, the contents of the low-order 8 bits (DBF1, DBF0) of the data buffer DBF are transferred to the peripheral circuit (the contents of DBF3 and DBF2 are not transferred).

5.33 SKT m, #n Skip next instruction if data memory bits are true

① Operation code

11110	m_H	m_L	n

② Function

CMP ← 0, if (M) AND n = n, then skip

Skips the next instruction if the logical product of the contents of the data memory addressed by M and the immediate data n is not 0.

③ Example 1

To jump to AAA if bit 0 of address 0.03H is "1", or jump to BBB if "0".

```
MEM003  MEM   0.03H
        MOV   BANK, #00H       ; Data memory bank set to 0
        SKT   MEM003, #0001B
        BR    BBB
        BR    AAA
```

Example 2

To skip the next instruction if both bit 0 and bit 1 of address 0.03H are "1".

```
MEM003  MEM   0.03H
        MOV   BANK, #00H       ; Data memory bank set to 0
        SKT   MEM003, #0011B
```

Skip Condition 03H

B_3	B_2	B_1	B_0
×	×	1	1

× ; don't care

Example 3

The result of executing the following two instructions is identical.

```
MEM013  MEM   0.13H
        SKT   MEM013, #1111B
        SKE   MEM013, #0FH
```

5.34 SKF m, #n Skip next instruction if data memory bits are false

① Operation code

11111	m_H	m_L	n

② Function

CMP ← 0, if (M) AND n = 0, then skip

Skips the next instruction if the logical product of the contents of the data memory addressed by M and the immediate data n is 0.

③ Example 1

This program stores the immediate data 00H in data memory address 0.0FH if bit 2 of address 0.13H is "0", or jumps to ABC if "1".

```
MEM013  MEM   0.13H
MEM00F  MEM   0.0FH
        MOV   BANK, #00H       ; Data memory bank set to 0
        SKF   MEM013, #0100B
        BR    ABC
        MOV   MEM00F, #00H
```

Example 2

To skip the next instruction if both bit 3 and bit 0 of address 0.29H are "0".

```
MEM029  MEM   0.29H
        MOV   BANK, #00H       ; Data memory bank set to 0
        SKF   MEM029, #1001B
```

		B_3	B_2	B_1	B_0
Skip Condition	29H	0	×	×	0

× : don't care

Example 3

The result of executing the following two instructions is identical.

```
MEM034  MEM   0.34H
        SKF   MEM034, #1111B
        SKE   MEM034, #00H
```

5.35 BR addr — Branch to the address

① Operation code

01100	addr
01101	
01110	
01111	

② Function

if branch to page0, PAGE ← 0, PC (10-0) ← addr

if branch to page1, PAGE ← 1, PC (10-0) ← addr

if branch to page2, PAGE ← 2, PC (10-0) ← addr

if branch to page3, PAGE ← 3, PC (10-0) ← addr

Branches to the address indicated by addr.

The address range to which this instruction can branch directly comprises the 8K steps from address 0000H to address 1FFFH.

To branch to address 2000H or above, the "BR @AR" instruction described below should be used.

③ Example

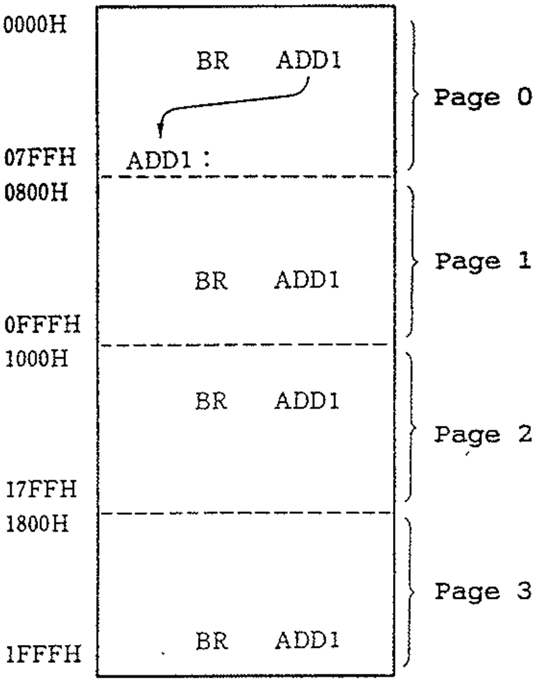
```
FLY     LAB     0FH          ; Define FLY = OFH.
         :
         :
         :
        BR      FLY          ; Branch to address OF.
         :
         :
         :
        BR      LOOP1        ; Branch to LOOP1.
         :
         :
         :
        BR      $ + 2        ; Jump to 2 addresses below current
                               address.
         :
         :
         :
        BR      $ - 3        ; Jump to 3 addresses above current
                               address.
         :
         :
         :
LOOP1 :
```

④ Note

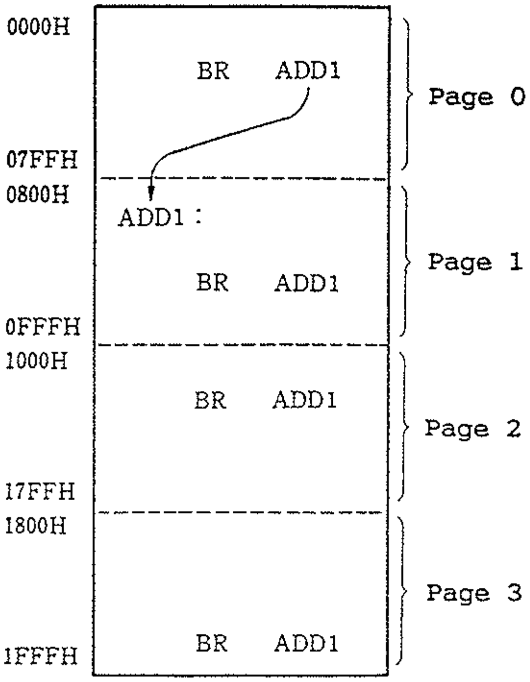
The operation codes for a direct branch instruction to inner page 0, inner page 1, inner page 2 and inner page 3 are different from each other. The operation codes for a direct branch instruction to inner page 0, inner page 1, inner page 2 and inner page 3 are 'OCH', 'ODH', 'OEH' and 'OFH', respectively.

This is because direct branch instruction operand addr is 11 bits and uses the operation code low-order 2 bits as an address of the branch destination. When the 17K series assembler (AS17K) checks these operation codes and references the branch destination specified by a label, the operation code bits are automatically converted to the branch destination page address.

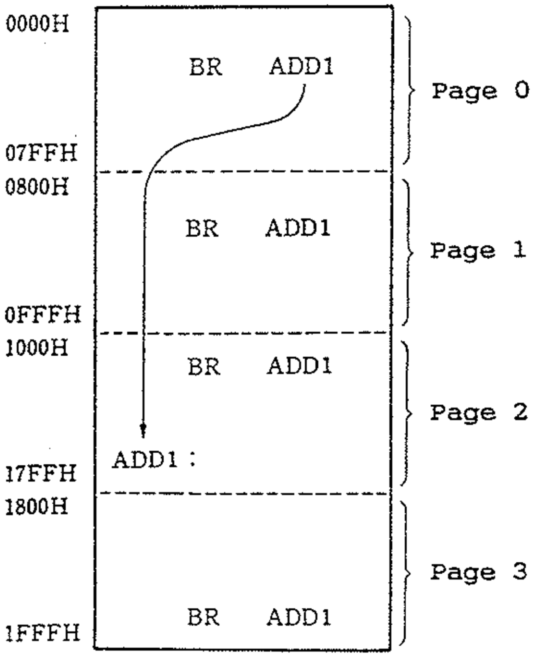
Operation Code = 0C
(Branch Destination
Address in Page 0)



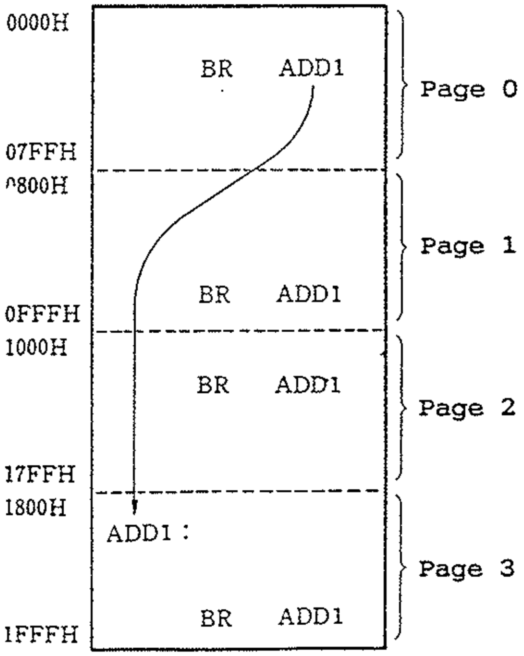
Operation Code = 0D
(Branch Destination
Address in Page 1)



Operation Code = 0E
(Jump Destination
Address in Page 2)

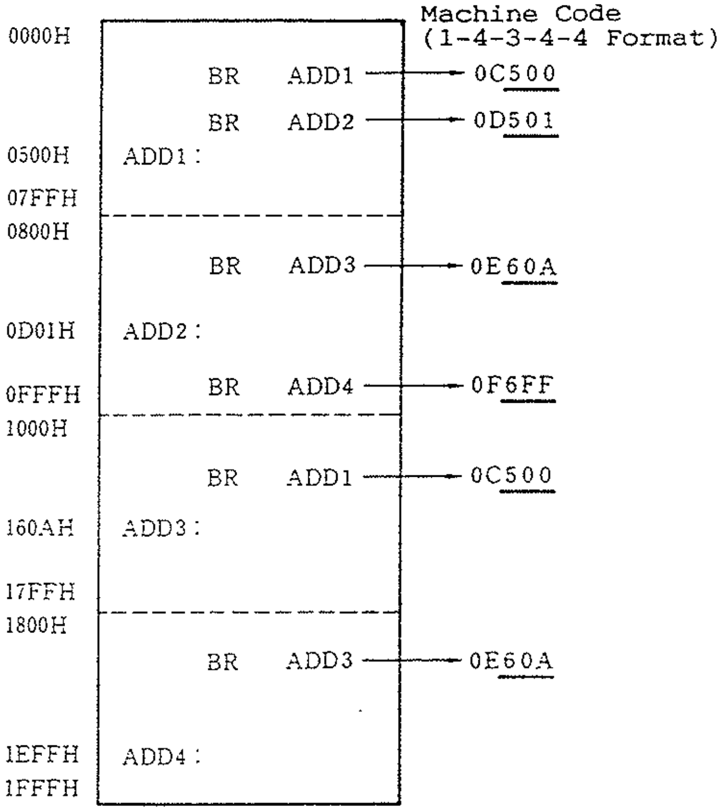


Operation Code = 0F
(Jump Destination
Address in Page 3)



When patch corrections are made during debugging, conversion of "0C", "0D", "0E" and "0F" must be performed by a user.

Also, address translation is necessary when the branch destination of the BR instruction is in the address range 0000H to 07FFH, 0800H to 0FFFH, 1000H to 17FFH, and 1800H to 1FFFH respectively. That is, address 0000H, 0800H, 1000H or 1800H is made address 000H and subsequent addresses are each incremented by one.



The number of pages varies from product to product in the 17K series: Refer to the Data Sheet for the product used.

5.36 BR @AR Branch to the address specified by address register

① Operation code

00111	000	0100	0000

② Function

PC ← (AR)

Branches to the address indicated by the address register (AR).

③ Example 1

To set 003FH in the address register AR (AR0 to AR3), and jump to address 003FH using the "BR @AR" instruction.

```
MOV     AR3,  #00H     ; AR3 ← 00H
MOV     AR2,  #00H     ; AR2 ← 00H
MOV     AR1,  #03H     ; AR1 ← 03H
MOV     AR0,  #0FH     ; AR0 ← 0FH
BR      @AR            ; Jumps to address 003FH
```

Example 2

To change the branch destination as shown below according to the contents of data memory address 0.10H.

0.10H Contents		Branch Destination Label
00H	→	AAA
01H	→	BBB
02H	→	CCC
03H	→	DDD
04H	→	EEE
05H	→	FFF
06H	→	GGG
07H	→	HHH
08H-0FH	→	ZZZ

```
                 ; *
                 ; * * Jump table
Address          ; *
 0010H           BR          AAA
 0011H           BR          BBB
 0012H           BR          CCC
 0013H           BR          DDD
 0014H           BR          EEE
 0015H           BR          FFF
 0016H           BR          GGG
 0017H           BR          HHH
 0018H           BR          ZZZ
                                     :
                                     :
                                     :
                                     :
                                     :
 MEM010          MEM         0.10H
                 MOV         RPH,    #00H    ; General register bank set to 0
                 MOV         RPL,    #02H    ; General register row address set to 1
                 MOV         AR3,    #00H    ;AR3←00H AR set to 001xH
                 MOV         AR2,    #00H    ;AR2←00H
                 MOV         AR1,    #01H    ;AR1←01H
```

```
ST      AR0,    MEM010  ; AR0 ← 0.10H
SKF     AR0,    #1000B  ; If AR0 contents are greater than 08H,
                          AR0 contents are set to 08H
AND     AR0,    #1000B  ;
BR      @AR
```

④ Note

The number of bits which can be used in the address register (AR3, AR2, AR1, AR0) varies from product to product: Please refer to the Data Sheet for the product concerned when using this register.

5.37 RORC r Rotate right general register with carry flag

① Operation code

00111	000	0111	r

② Function

(CY) → R3 → R2 → R1 → R0 → CY

Shifts the contents of the general register indicated by R including the carry flag one bit to the right.

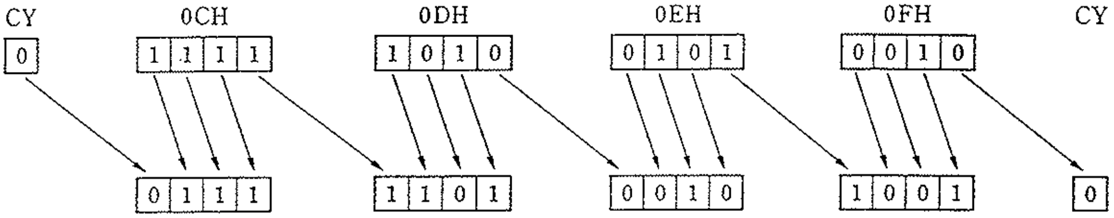
③ Example 1

When bank 0 row address 0 (0.00H to 0.0FH) is specified as general registers (RPH = 0, RPL = 0), this program shifts the value of address 0.00H (1000B) one bit to the right, giving 0100B.

```
            0.00H ←  (0.00H) ÷ 2
MEM000  MEM      0.00H
        MOV      RPH, #00H  ; General register bank set to 0
        MOV      RPL, #00H  ; General register row address set to 0
        CLR1     CY         ; Carry flag ← 0
        RORC     MEM000
```

Example 2

When bank 0 row address 0 (0.00H to 0.0FH) is specified as general registers (RPH = 0, RPL = 0), this program shifts the data buffer (DBF) value 0FA52H one bit to the right, giving 7D29H.



```
MOV   RPH, #00H  ; General register bank set to 0
MOV   RPL, #00H  ; General register row address set to 0
CLR1  CY         ; Carry flag ← 0
RORC  DBF3
RORC  DBF2
RORC  DBF1
RORC  DBF0
```

5.38 CALL addr — Call subroutine

① Operation code

11100	addr

② Function

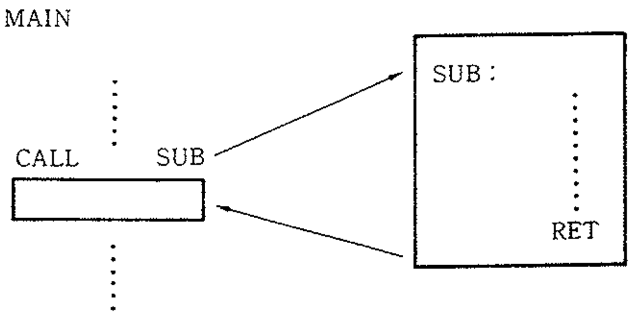
```
SP ← (SP) - 1,
ASR ← PC + 1,
PAGE ← 0,
PC (10 to 0) ← addr
```

Increments the program counter (PC) value and stores it in the stack, then branches to the subroutine indicated by addr.

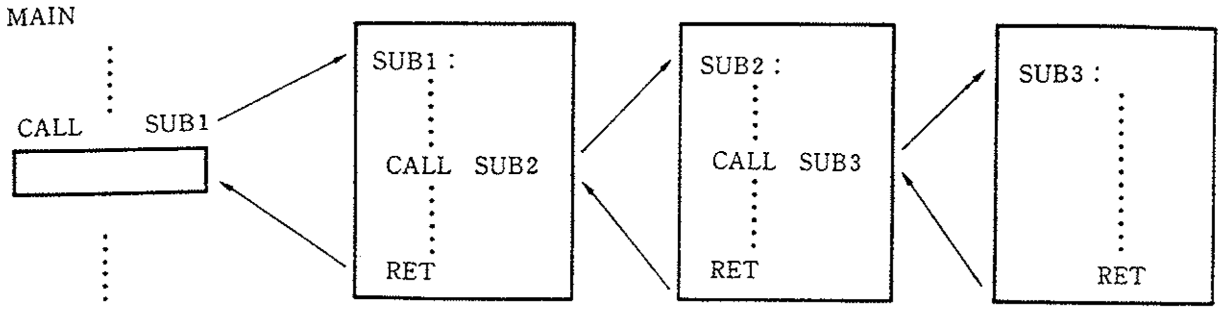
The subroutine called by this instruction must be within the 2K steps from address 0000H to address 07FFH. It is therefore useful to locate frequently used subroutines within the range from address 0000H to address 07FFH.

To call a subroutine located in address 0800H onward, the "CALL @AR" instruction described next should be used.

③ Example 1



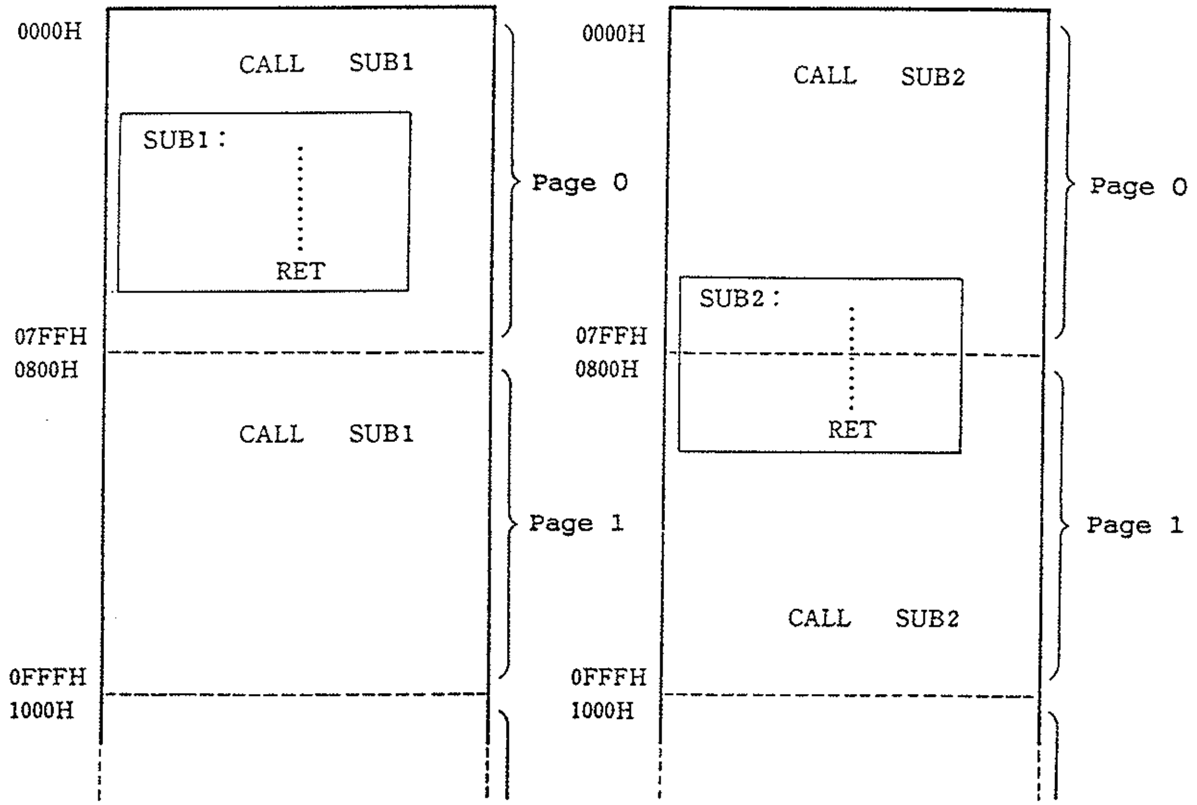
Example 2



④ Note

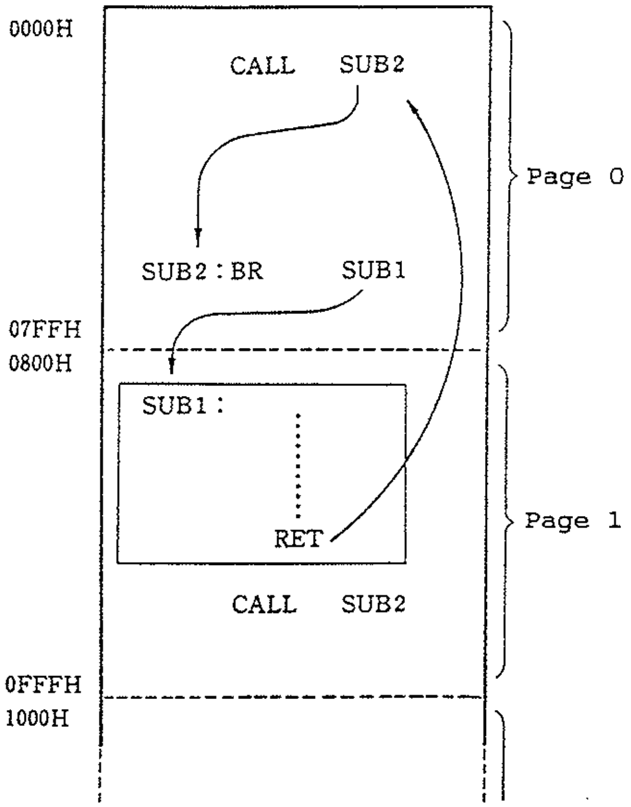
When the "CALL" instruction is used, the address called, that is the subroutine start address, must be located in page 0 (addresses 0000H to 07FFH). To call a subroutine with a start address outside page 0, the "CALL @AR" instruction should be used.

Example with Subroutine Start Address in Page 0



If the subroutine start address is in page 0, as shown above, it does not matter if the subroutine end address ("RET" or "RETSK" instruction) is outside page 0.

As long as the subroutine start address is in page 0, the "CALL" instruction can be used without regard to the page concept. If it is not possible to locate the subroutine start address in page 0 when writing the program, the following method can be used to advantage.



Here, a "BR" instruction is placed in page 0, and the actual subroutine is called via this "BR" instruction.

5.39 CALL @AR Call subroutine specified by address register

① Operation code

00111	000	0101	0000

② Function

SP ← (SP) - 1,
ASR ← PC + 1,
PC ← (AR)

Increments the program counter (PC) value and stores it in the stack, then branches to the subroutine indicated by the address register (AR).

③ Example 1

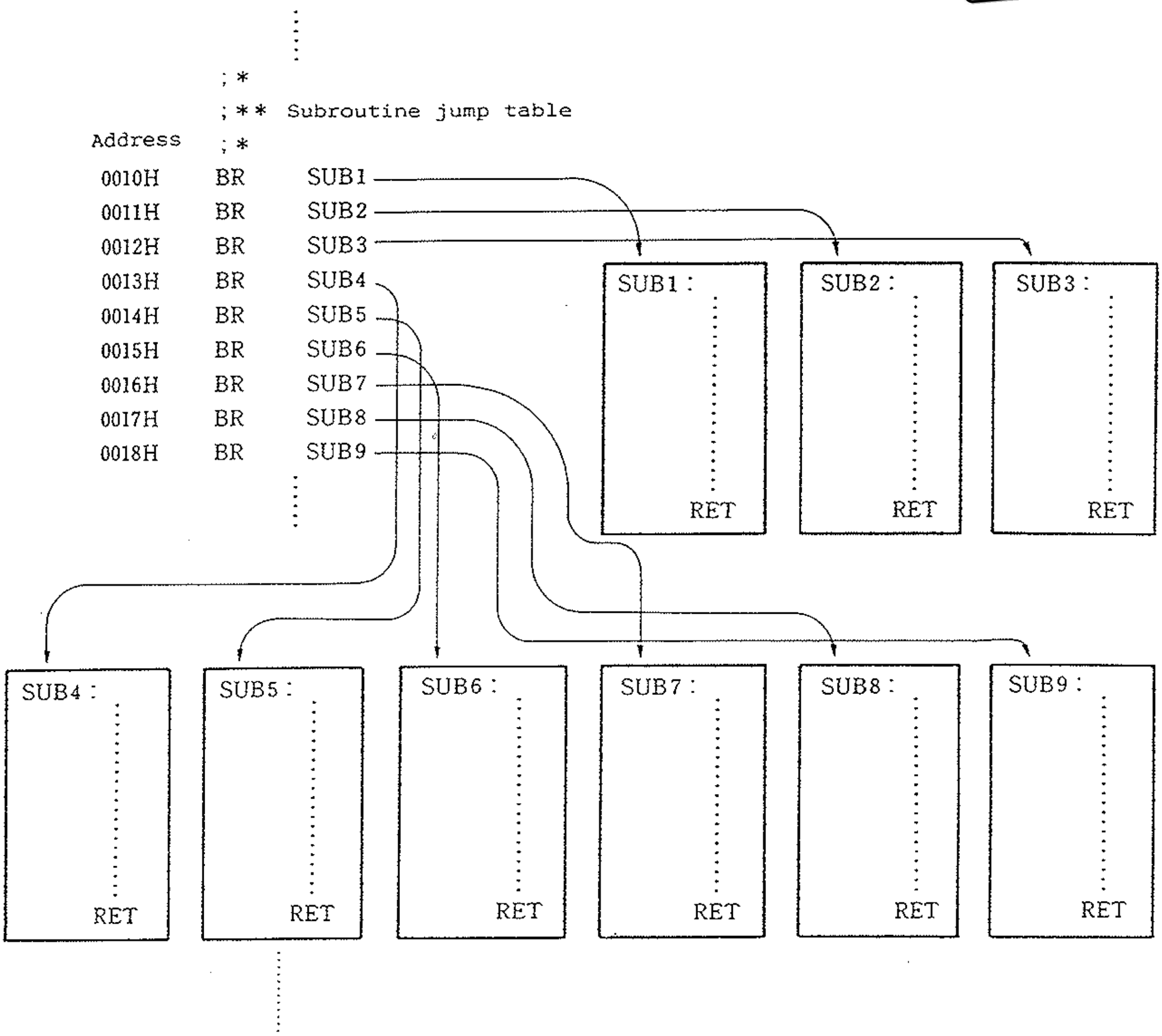
To set 0020H in the address register AR (AR0 to AR3), and call the subroutine at address 0020H by means of the "CALL @AR" instruction.

```
MOV     AR3, #00H     ; AR3 ← 00H
MOV     AR2, #00H     ; AR2 ← 00H
MOV     AR1, #02H     ; AR1 ← 02H
MOV     AR0, #00H     ; AR0 ← 00H
CALL    @AR           ; Calls subroutine at address 0020H
```

Example 2

To call the subroutines shown below according to the contents of data memory address 0.10H.

0.10H Contents		Subroutine Name
00H	→	SUB1
01H	→	SUB2
02H	→	SUB3
03H	→	SUB4
04H	→	SUB5
05H	→	SUB6
06H	→	SUB7
07H	→	SUB8
08H-0FH	→	SUB9



```
                  :
          ;*
          ;**  Subroutine jump table
Address   ;*
 0010H     BR     SUB1
 0011H     BR     SUB2
 0012H     BR     SUB3
 0013H     BR     SUB4
 0014H     BR     SUB5
 0015H     BR     SUB6
 0016H     BR     SUB7
 0017H     BR     SUB8
 0018H     BR     SUB9
                  :
```

```
MEM010  MEM    0.10H
        MOV    RPH,  #00H     ; General register bank set to 0
        MOV    RPL,  #02H     ; General register row address set to 1
        MOV    AR3,  #00H     ; AR3 ← 00H  Address register set to 001xH
        MOV    AR2,  #00H     ; AR2 ← 00H
        MOV    AR1,  #01H     ; AR1 ← 01H
        ST     AR0,  MEM010   ; AR0 ← 0.10H
        SKF    AR0,  #1000B   ;} If AR0 contents are greater than 08H, AR0
        AND    AR0,  #1000B   ;} contents are set to 08H
        CALL   @AR    ──────────────→  To jump table
               :      ←─────────────
               :                       Return here upon execution
               :                       of "RET" instruction by
                                       respective subroutine.
```

④ Note

The number of bits which can be used in the address register (AR3 to AR0) varies according to the type of device: Refer to the device manual when using this register.

5.40 SYSCAL entry   Call system segment entry address

① Operation code

00111	$entry_H$	0000	$entry_L$

② Function

$SP \leftarrow (SP) - 1$, $ASR \leftarrow PC + 1$,

$SGR \leftarrow SYSSEG$, $PAGE \leftarrow 0$, $PC\ (10\ \text{to}\ 8) \leftarrow entry_H$,

$PC\ (7\ \text{to}\ 4) \leftarrow 0$, $PC\ (3\ \text{to}\ 0) \leftarrow entry_L$

After incrementing the program counter (PC) value and storing it in the stack, this instruction branches it to the subroutine indicated by "entry" which is in system segment page 0.

The subroutine which can be called by this instruction is 256 steps of the system segment entry address which is in system segment page 0.

③ Example 1

```
MAIN :
            ⋮
            SYSCAL  34H
            ⋮
CSEG        n
ORG         304H
            ⋮
            RET
  (n:  System segment)
```

Example 2

```
MAIN :
          ⋮
          SYSCAL .DL.((ENTRY SHR 4 AND 0070H)OR(ENTRY AND 000FH))
          ⋮
ENTRY :
          ⋮
          RET
```

④ Note

For the "SYSCAL" operand do not describe a label type symbol but a data type symbol. If the operand value exceeds 7 bits, the assembler (AS17K) generates an error.

In Example 2 of ③ above, even though the "ENTRY" address is not in the system segment entry address, the assembler does not generate an error. In this case, the branch destination by the "SYSCAL" instruction is different from the address for which the user intended and therefore special care is required in debugging.

5.41 RET Return to the main program from subroutine

① Operation code

00111	000	1110	0000

② Function

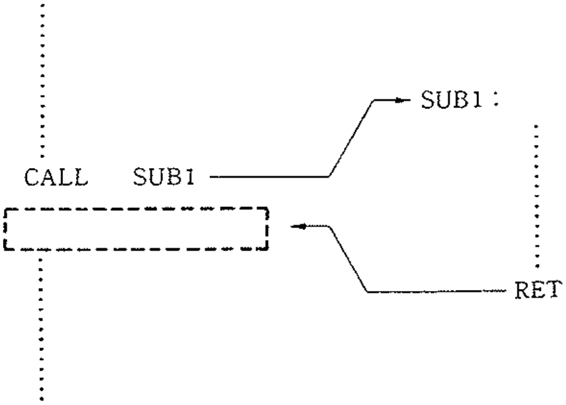
PC ← (ASR),

SP ← (SP) + 1

This instruction is used to return to the main program from a subroutine.

The return address saved to the stack by the CALL instruction is restored to the program counter.

③ Example



5.42 RETSK Return to the main program then skip next instruction

① Operation code

00111	001	1110	0000

② Function

PC ← (ASR),
SP ← (SP) + 1,
and skip

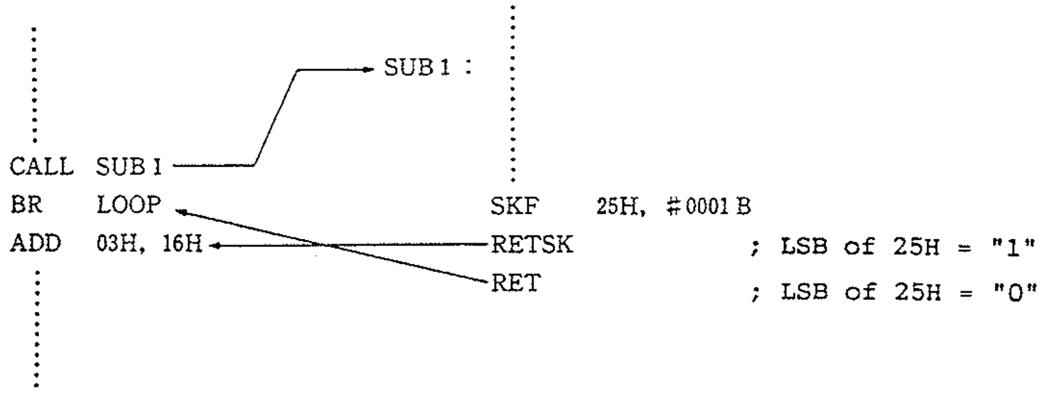
This instruction is used to return to the main program from a subroutine.

The instruction following the "CALL" instruction is skipped.

That is, the program counter (PC) is incremented after the return address saved to the stack by the CALL instruction is restored to the program counter.

③ Example

In this example, when the value of the LSB (least significant bit) of data memory (RAM) address 25H is "0", a "RET" instruction is executed and control returns to the instruction following the "CALL" instruction, and when "1", a "RETSK" instruction is executed and control returns to the instruction following the instruction after the "CALL" instruction (in this case ADD 03H, 16H).



```
                          SUB1 :
   :                           :
   :                           :
CALL   SUB1                    :
BR     LOOP               SKF      25H, #0001B
ADD    03H, 16H           RETSK                  ; LSB of 25H = "1"
   :                      RET                    ; LSB of 25H = "0"
   :
```

5.43 RETI Return to the main program from interrupt service routine

① Operation code

00111	100	1110	0000

② Function

PC ← (ASR),
SP ← (SP) + 1,
INTR ← (INTSK)

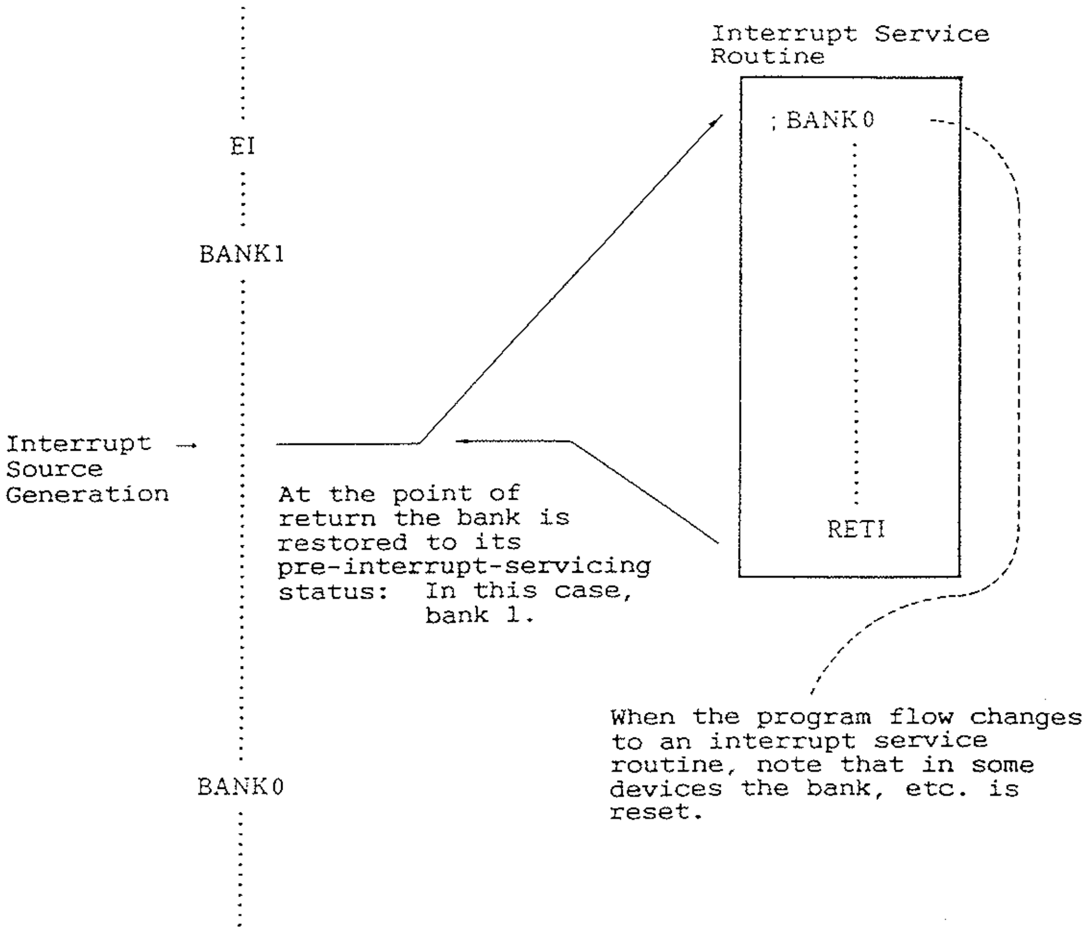
This instruction is used to return to the main program from an interrupt service routine.

The return address saved to the stack by the vectored interrupt is restored to the program counter.

Depending on the device, some system registers may also be restored to their status prior to generation of the vectored interrupt.

③ Example

In this example a vectored interrupt is generated when the data memory is in bank 1, and since data memory bank 0 is required for interrupt servicing the bank must be saved.



At the point of return the bank is restored to its pre-interrupt-servicing status: In this case, bank 1.

When the program flow changes to an interrupt service routine, note that in some devices the bank, etc. is reset.

④ Note 1

The system register contents automatically saved by an interrupt (and restorable by the "RETI" instruction) vary depending on the product: Please take special care in consulting the relevant Data Sheet.

Note 2

If the "RETI" instruction is used instead of the "RET" instruction in a normal subroutine, the bank etc. (items saved by the interrupt) are restored upon return to the return address, and what kind of state is set is undefined. Therefore, the "RET" (or "RETSK") instruction must always be used to return from a subroutine.

5.44 EI enable Interrupt

① Operation code

00111	000	1111	0000

② Function

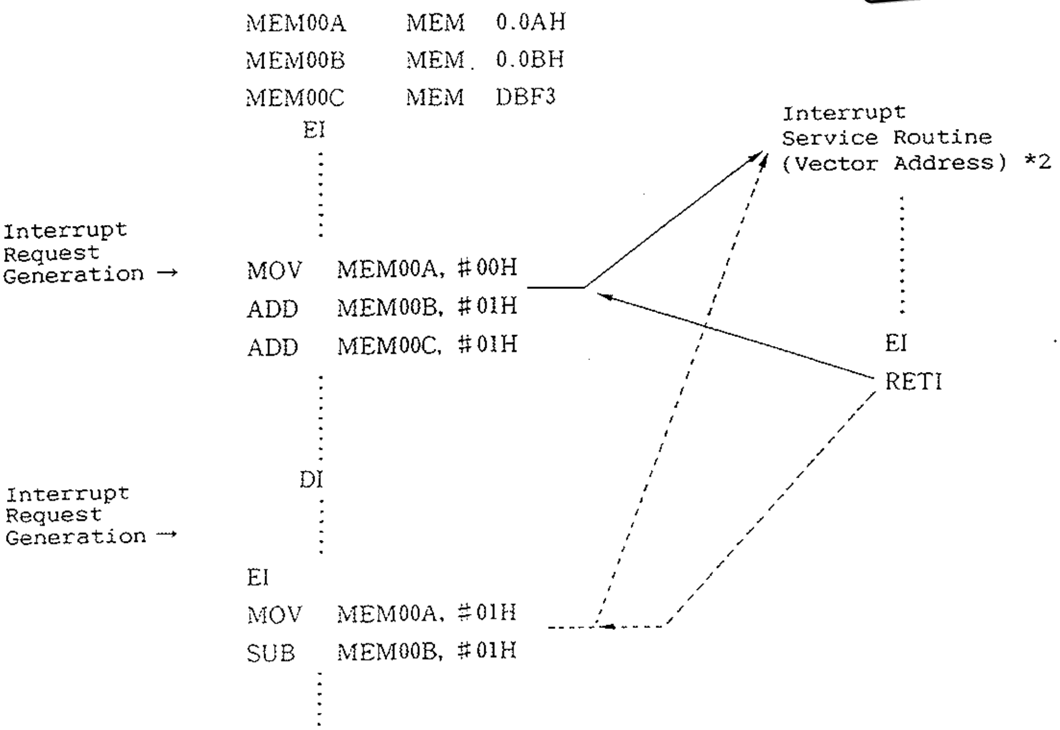
INTEF ← 1

Enables vectored interrupts.

Interrupts are enabled after execution of the instruction following the "EI" instruction.

③ Example 1

As can be seen from the following example, when an interrupt request is acknowledged the flow changes to the vector address after execution of the instruction (excluding an instruction which manipulates the program counter) following acknowledgment is complete.*1



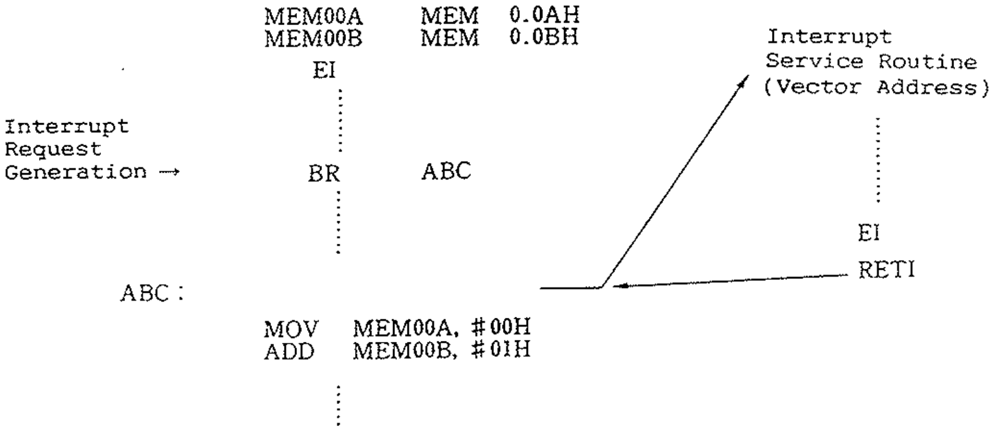
```
                                MEM00A     MEM   0.0AH
                                MEM00B     MEM.  0.0BH
                                MEM00C     MEM   DBF3
                                   EI                            Interrupt
                                    :                            Service Routine
                                    :                            (Vector Address) *2
Interrupt                                                              :
Request                                                                :
Generation →                    MOV    MEM00A, #00H                    :
                                ADD    MEM00B, #01H
                                ADD    MEM00C, #01H                  EI
                                    :                                RETI
                                    :
                                   DI
Interrupt                           :
Request                             :
Generation →                    EI
                                MOV    MEM00A, #01H
                                SUB    MEM00B, #01H
                                    :
                                    :
```

*1: The vector address depends on the interrupt acknowledged: Refer to the Data Sheet for the product used for details.

2: Interrupts which can be acknowledged here (for which an interrupt is generated after execution of the EI instruction, followed by a change of flow to an interrupt service routine) are those for which the corresponding interrupt permission flag (IPxxx) is set. An interrupt request generated after execution of the EI instruction when no interrupt permission flags are set will not result in a change in the program flow (the interrupt will not be acknowledged). However, since the interrupt request flag (IRQxxx) is set, the interrupt will be acknowledged as soon as the interrupt permission flag is set. (See the Data Sheet for the relevant product for details.)

Example 2

In the following example an interrupt is generated by an interrupt request acknowledged during execution of an instruction which manipulates the program counter (PC).



```
            MEM00A    MEM   0.0AH
            MEM00B    MEM   0.0BH
              EI
               :
               BR      ABC
               :
     ABC:
            MOV     MEM00A, #00H
            ADD     MEM00B, #01H
               :
```

```
Interrupt Service Routine (Vector Address)
      :
     EI
     RETI
```

5.45 DI Disable interrupt

① Operation code

00111	001	1111	0000

② Function

INTEF ← 0

Disables vectored interrupts.

③ Example

See Example 1 under 5.44 "EI".

④ Note 1

Executing the "DI" instruction sets an interrupt acknowledge disable (DI) state and no program flow is changed even though an interrupt request is generated. However, an interrupt request flag (IRQxxx) is set. Therefore, by executing the "EI" instruction, immediately after execution of an instruction following the "EI" instruction, an interrupt is acknowledged and the program flow is shifted to the interrupt vector address. In this case, it is necessary that the corresponding interrupt enable flag (IPxxx) has been set.

Note 2

An interrupt is acknowledged without execution the "DI" instruction and when the program flow is shifted to the interrupt vector address, the DI state is automatically set. Therefore, ensure that the "EI" instruction is executed immediately before execution of the "RETI" instruction returned from the interrupt servicing in order to set such software that always enables interrupt acknowledgment in the main routine.

5.46 STOP s — Stop CPU and release by condition s

① Operation code

00111	010	1111	s

② Function

stop clock

Stops the main clock and places the device into the STOP mode.

Placing the device into the STOP mode enables the consumption current to be kept to a minimum.

The condition for releasing the STOP mode and starting main clock oscillation is specified by the operand "s".

The stop release condition "s" varies from product to product: Please refer to the Data Sheet for the product concerned when using this instruction.

5.47 HALT h — Halt CPU and release by condition h

① Operation code

00111	011	1111	h

② Function

halt CPU

Places the device into the HALT mode. In the HALT mode, the CPU executes no instruction.

Placing the device into the HALT mode enables the consumption current to be reduced.

The condition for releasing the HALT mode is specified by the operand "h".

The halt release condition "h" varies from product to product: Please refer to the Data Sheet for the product concerned when using this instruction.

5.48 NOP No operation

① Operation code

00111	100	1111	0000

② Function

no operation

Expends one machine cycle without performing any operation.

APPENDIX. INDEX OF INSTRUCTIONS

Phase-out/Discontinued

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: ____________________

Document number: ____________________ Page number: __________

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	□	□	□	□
Technical Accuracy	□	□	□	□
Organization	□	□	□	□