

RL78 Family

User's Manual: Software

Single-Chip Microcontrollers

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (<http://www.renesas.com>).

Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

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

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

Readers	This manual is intended for users who wish to understand the functions of RL78 microcontrollers and to design and develop its application systems and programs.
Purpose	This manual is intended to give users an understanding of the various kinds of instruction functions of RL78 microcontrollers.
Organization	This manual is broadly divided into the following sections. • CPU functions • Instruction set • Explanation of instructions
How to Read This Manual	It is assumed that readers of this manual have general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers. • To check the details of the functions of an instruction whose mnemonic is known: → Refer to Appendix A Instruction Index (Mnemonic by Function) and Appendix B Instruction Index (Mnemonic in Alphabetical Order). • To check an instruction whose mnemonic is not known but whose general function is known: → Find the mnemonic in Section 5 Instruction Set and then check the detailed functions in Section 6 Explanation of Instructions. • To learn about the various kinds of RL78 microcontroller instructions in general: → Read this manual in the order of CONTENTS. • To learn about the hardware functions of RL78 microcontrollers: → See the user's manual for each microcontroller.
Conventions	Data significance: Higher digits on the left and lower digits on the right Note: Footnote for item marked with Note in the text Caution: Information requiring particular attention Remark: Supplementary information Numerical representations: Binary numbers are represented as xxxx or xxxxB, where each x is 0 or 1. Decimal numbers are represented as xxxx, where each x is a numeral from 0 to 9. Hexadecimal numbers are represented as xxxxH, where each x is a number from 0 to 9 or a letter from A to F.

CONTENTS

1.	Overview	1
1.1	Features	1
1.2	Differences in Functions of the Three CPU Cores	1
2.	Memory Space	2
2.1	Memory Space	2
2.2	Internal Program Memory Space	3
2.2.1	Mirror area	3
2.2.2	Vector table area	4
2.2.3	CALLT instruction table area	4
2.2.4	Option bytes area	5
2.2.5	Security ID setting area	5
2.3	Internal Data Memory (Internal RAM) Space	5
2.4	Special Function Register (SFR) Area	5
2.5	Extended Special Function Register (2nd SFR) Area	6
3.	Registers	7
3.1	Control Registers	7
3.1.1	Program counter (PC)	7
3.1.2	Program status word (PSW)	7
3.1.3	Stack pointer (SP)	9
3.2	General-purpose Registers	11
3.2.1	General-purpose registers of RL78-S1 core	11
3.2.2	General-purpose registers of RL78-S2 core and RL78-S3 core	12
3.3	ES and CS Registers	14
3.4	Special Function Registers (SFRs)	15
3.4.1	Processor mode control register (PMC)	15
4.	Addressing	17
4.1	Program Addressing	17
4.1.1	Relative addressing	17
4.1.2	Immediate addressing	18
4.1.3	Table indirect addressing	18
4.1.4	Register indirect addressing	19
4.2	Data Addressing	20
4.2.1	Implied addressing	20
4.2.2	Register addressing	20
4.2.3	Direct addressing	21
4.2.4	Short direct addressing	22
4.2.5	SFR addressing	23
4.2.6	Register indirect addressing	24
4.2.7	Based addressing	25
4.2.8	Based indexed addressing	29
4.2.9	Stack addressing	31

5.	Instruction Set	35
5.1	Operand Identifiers and Description Methods	35
5.2	Description in “Operation” Column	37
5.3	Description in “Flag” Column	38
5.4	PREFIX Instruction	38
5.5	List of Operations	39
5.5.1	Operation list of RL78-S1 core	39
5.5.2	Operation list of RL78-S2 core	56
5.5.3	Operation list of RL78-S3 core	73
5.6	Instruction Format	91
5.7	Instruction Maps	123
6.	Explanation of Instructions	128
6.1	8-bit Data Transfer Instructions	129
6.2	16-bit Data Transfer Instructions	136
6.3	8-bit Operation Instructions	142
6.4	16-bit Operation Instructions	153
6.5	Multiply, Divide, and Multiply-and-accumulate Instructions	157
6.6	Increment and Decrement Instructions	165
6.7	Shift Instructions	170
6.8	Rotate Instructions	177
6.9	Bit Manipulation Instructions	183
6.10	Call and Return Instructions	191
6.11	Stack Manipulation Instructions	198
6.12	Unconditional Branch Instruction	204
6.13	Conditional Branch Instructions	206
6.14	Conditional Skip Instructions	216
6.15	CPU Control Instructions	223
7.	Pipeline	230
7.1	Features	230
7.2	Numbers of Operating Clock Cycles	231
7.2.1	Access to flash memory contents as data	231
7.2.2	Fetching instructions from RAM	231
7.2.3	Hazards related to combinations of instructions	232
Appendix A	Instruction Index (Mnemonic by Function)	233
Appendix B	Instruction Index (Mnemonic in Alphabetical Order)	236
Appendix C	Revision History	239
C.1	Major Revisions in This Edition	239
C.2	Major Revisions in This Edition	240

Section 1 Overview

The CPU core in the RL78 microcontroller employs the Harvard architecture which has independent instruction fetch bus, address bus and data bus. Through the adoption of three-stage pipeline control over fetching, decoding, and memory access, the efficiency of operation is markedly improved over that of the previous (S2) CPU core. The CPU core features high performance and highly-functional instruction processing, and is suitable for use in various applications that require high-speed and high-functionality processing.

1.1 Features

The CPU cores of the RL78 microcontrollers are classified into three types (RL78-S1, RL78-S2, and RL78-S3) based on differences in the number of instructions, clock cycles, and performance. The main features of the RL78 CPU cores are listed below.

- 3-stage pipeline CISC architecture
- Address space: 1 Mbyte
- Minimum instruction execution time: One instruction execution per clock cycle
- Eight 8-bit general-purpose registers
- Number of instruction mnemonics: 74 (RL78-S1 core), 75 (RL78-S2 core), 81 (RL78-S3 core)
- Data allocation: Little endian

1.2 Differences in Functions of the Three CPU Cores

Table 1 - 1 shows the differences in functions of the RL78-S1, RL78-S2, and RL78-S3 cores.

Table 1 - 1 Differences in Functions of the Three CPU Cores

Parameter	RL78-S1 Core	RL78-S2 Core	RL78-S3 Core
CPU	8 bits	16 bits	16 bits
Number of instruction mnemonics	74	75	81
General-purpose registers	Eight 8-bit registers (no banks)	Four banks of eight 8-bit registers	Four banks of eight 8-bit registers
Multiply/divide/multiply-and-accumulate instruction	Not provided	Not provided	Provided

Caution The instructions are functionally the same across all three CPU cores. However, some instructions require a different number of clock cycles on the RL78-S1 core than on the other cores. For details, see 5.5 List of Operations.

Remark The CPU core depends on the product. The following examples show products that use each CPU core. For product groups other than those listed, refer to the user's manual for the given product.

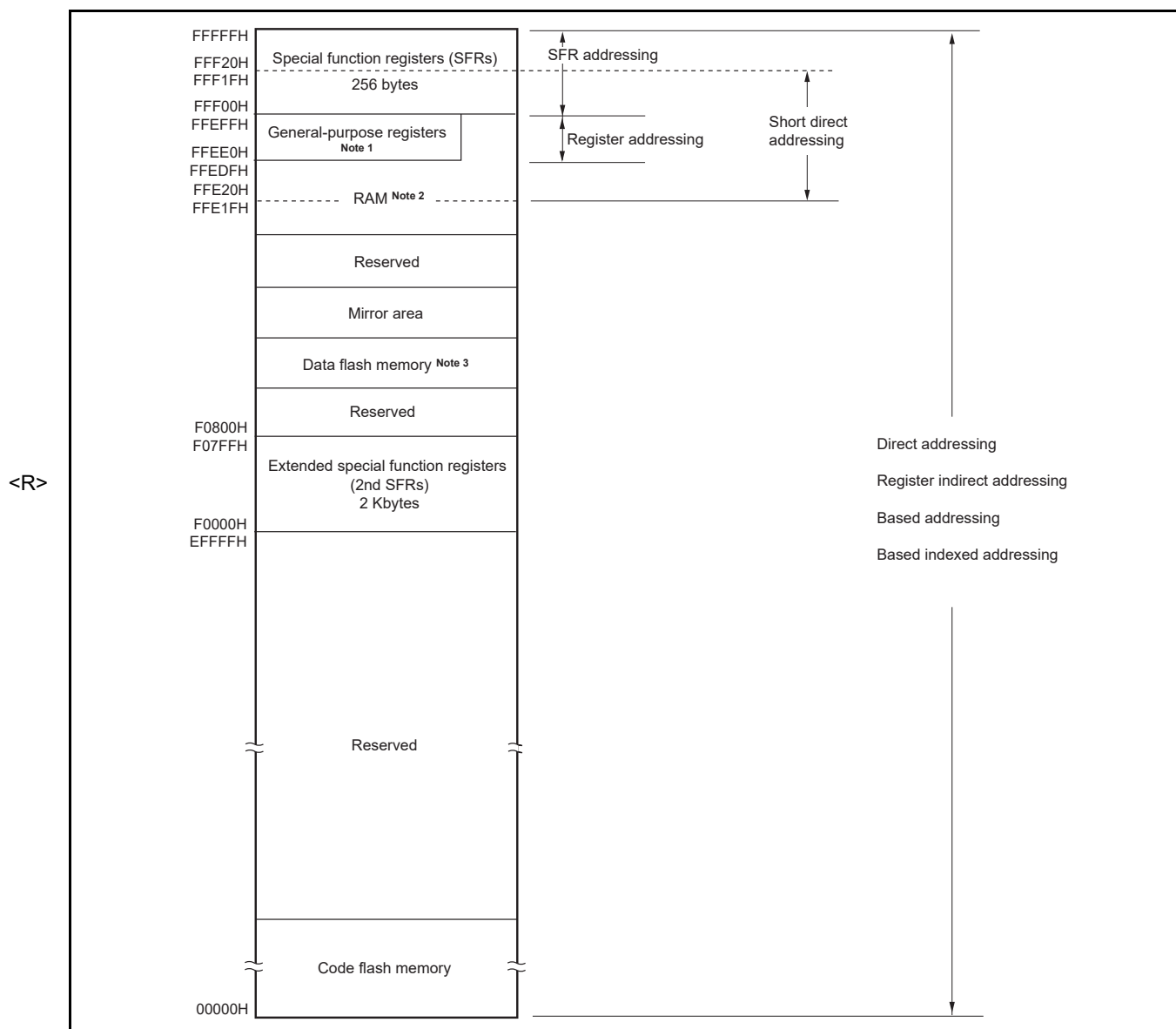
- RL78-S1 core: RL78/G10, RL78/G1M, RL78/G1N
- RL78-S2 core: RL78/G12, RL78/G13, RL78/G13A, RL78/G15, RL78/G16, RL78/G1A, RL78/G1C, RL78/G1D, RL78/G1E, RL78/G1P, RL78/I1A, RL78/L12, RL78/L13, RL78/D1A, RL78/F12
- RL78-S3 core: RL78/G21, RL78/G22, RL78/G23, RL78/G24, RL78/G11, RL78/G14, RL78/G1F, RL78/G1G, RL78/G1H, RL78/H1D, RL78/I1B, RL78/I1C, RL78/I1D, RL78/I1E, RL78/L23, RL78/L1A, RL78/L1C, RL78/F22, RL78/F23, RL78/F24, RL78/F25, RL78/F13, RL78/F14, RL78/F15

Section 2 Memory Space

2.1 Memory Space

A 1-Mbyte address space is accessible by an RL78 microcontroller. **Figure 2 - 1** shows the memory map of the microcontroller.

Figure 2 - 1 Memory Map of the RL78 Microcontroller



Note 1. The 8-byte area from FFEF8H to FFEFFH is reserved as the general-purpose register area of the RL78-S1 core. The area from FFE00H to FFEF7H is reserved. The 32-byte area from FFE00H to FFEFFH is reserved as the general-purpose register area of the RL78-S2 and RL78-S3 cores.

Note 2. Using parts of this area during self-programming or rewriting of the data flash memory is prohibited, since some of this area will be in use by each of the given libraries. The areas where usage is prohibited depend on the product. For details, refer to the user's manual for the product in use.

Note 3. The area is reserved in products which do not include data flash memory. Otherwise, the size of the data flash memory depends on the product. For details, refer to the user's manual for the product in use.

2.2 Internal Program Memory Space

In RL78 microcontrollers, the internal program memory space range is from 00000H to EFFFFH. For the maximum size of the internal ROM (flash memory), refer to the user's manual for the product in use.

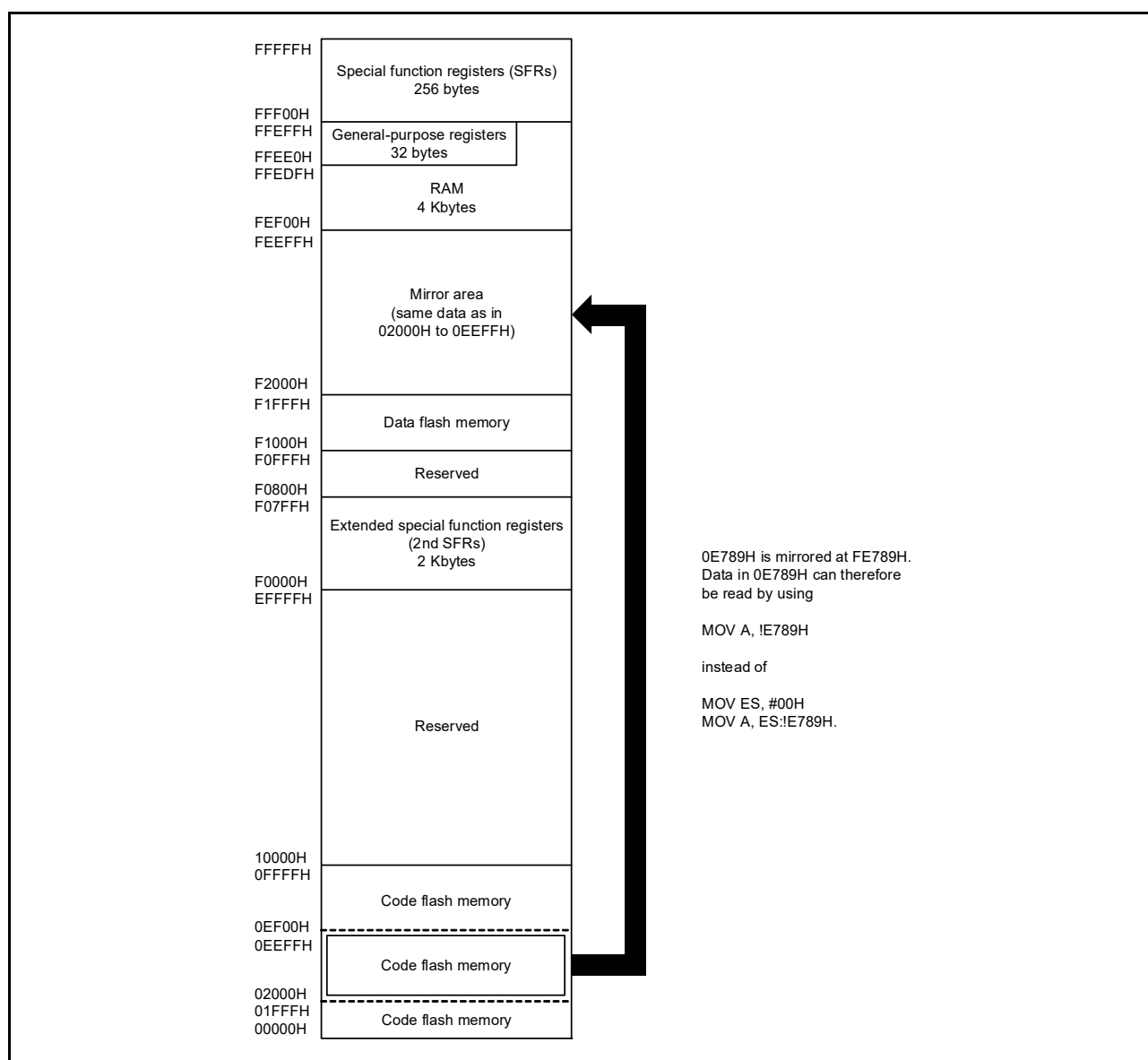
2.2.1 Mirror area

The mirror area depends on the CPU core as shown below. For details, see **3.4.1 Processor mode control register (PMC)**.

- RL78-S1 core
MAA = 0: Data in the address range from 00000H to 05EFFFH are mirrored in the range from F8000H to FDEFFFH.
MAA = 1: Setting prohibited
- RL78-S2 core
MAA = 0: Data in the address range from 00000H to 0FFFFH are mirrored in the range from F0000H to FFFFFH.
MAA = 1: Data in the address range from 10000H to 1FFFFH are mirrored in the range from F0000H to FFFFFH.
- RL78-S3 core
MAA = 0: Data in the address range from 00000H to 0FFFFH are mirrored in the range from F0000H to FFFFFH.
MAA = 1: Data in the address range from 10000H to 1FFFFH are mirrored in the range from F0000H to FFFFFH.

<R> Reading data from the mirrored area allows the use of instructions without the ES: prefix, so the contents of the code flash memory can be read with shorter instruction codes. However, the special function registers (SFRs), extended special function registers (2nd SFRs), RAM, data flash memory, and reserved areas are not mirrored. The mirror area depends on the product. Refer to the user's manual for the product in use. The mirror area is read-only. Instruction fetching from the mirror area is not possible. **Figure 2 - 2** shows an example.

Figure 2 - 2 Example of the RL78-S2 Core (Flash Memory: 64 Kbytes, RAM: 4 Kbytes)



2.2.2 Vector table area

The 128-byte area 00000H to 0007FH is reserved as a vector table area. The program start addresses for branch upon reset or generation of each interrupt request are stored in the vector table area. Furthermore, since 2-byte vector codes are assumed, the interrupt jump addresses are in the 64-Kbyte address range from 00000H to 0FFFFH. Of the 16-bit address, the lower 8 bits are stored at even addresses and the higher 8 bits are stored at odd addresses. When the boot swap function of a product which incorporates the RL78-S2 or RL78-S3 core is in use, a vector table area must also be set in boot cluster 1.

<R>

2.2.3 CALLT instruction table area

The 64-byte area 00080H to 000BFH can store the subroutine entry address of a 2-byte call instruction (CALLT). Set the subroutine entry addresses to values in the range from 00000H to 0FFFFH, since the address codes consist of 2 bytes. When the boot swap function of a product which incorporates the RL78-S2 or RL78-S3 core is in use, a CALLT instruction table area must also be set in boot cluster 1.

<R>

<R> 2.2.4 Option bytes area

The addresses for allocation and the number of bytes for the option bytes area depend on the product. Refer to the user's manual for the product in use. When the boot swap function of a product which incorporates the RL78-S2 or RL78-S3 core is in use, an option bytes area must also be set in boot cluster 1.

<R> 2.2.5 Security ID setting area

The security ID setting area serves as an on-chip debug security ID setting area or a flash programming security ID setting area^{Note}. When the boot swap function of a product which incorporates the RL78-S2 or RL78-S3 core is in use, a security ID setting area must also be set in boot cluster 1.

Note Security ID setting areas depend on the product. Refer to the user's manual for the product in use.

2.3 Internal Data Memory (Internal RAM) Space

The internal data memory (internal RAM) space can be used as a data area, except the area to which general-purpose registers are allocated, and a program area where instructions are written and executed. The higher limit of the address range of the RAM area is fixed to FFEFFH, and the range can be extended downward according to the amount of RAM in the product. The lower limit of the address range depends on the product in which the RAM is mounted. Refer to the user's manual for the product in use. The area to which general-purpose registers are allocated depends on the CPU core as shown below. For details, see **3.2 General-purpose Registers**.

- RL78-S1 core: FFEF8H to FFEFFH
- RL78-S2 core: FFEE0H to FFEFFH
- RL78-S3 core: FFEE0H to FFEFFH

Caution 1. Specify the stack area in an address range exclusive of that to which the general-purpose register area is assigned. The general-purpose register area cannot be used for fetching instructions or as a stack area.

Caution 2. Do not use relative addressing in instructions to branch from the RAM space to the internal program memory space.

2.4 Special Function Register (SFR) Area

Each of the special function registers (SFRs) is dedicated to a specific hardware control function, as opposed to the general-purpose registers. SFRs are allocated to the FFF00H to FFFFFH area. SFRs can be manipulated like general-purpose registers, using operation, transfer, and bit memory manipulation instructions. The manipulable bit units, 1, 8, and 16, depend on the SFR type. Manipulation bit unit can be specified as follows.

- 1-bit manipulation

Describe as follows for the 1-bit memory manipulation instruction operand (sfr.bit).

When the bit name is defined: <Bit name>

When the bit name is not defined: <Register name>, <Bit number> or <Address>, <Bit number>

- 8-bit manipulation

Describe the symbol defined by the assembler for the 8-bit memory manipulation instruction operand (sfr). This manipulation can also be specified with an address.

- 16-bit manipulation

Describe the symbol defined by the assembler for the 16-bit memory manipulation instruction operand (sfrp). When specifying an address, describe an even address.

2.5 Extended Special Function Register (2nd SFR) Area

Each of the extended special function registers (2nd SFRs) is dedicated to a specific hardware control function, as opposed to the general-purpose registers. 2nd SFRs are allocated to the F0000H to F07FFH area. SFRs other than those in the SFR area (FFF00H to FFFFFH) are allocated to this area. An instruction that accesses the 2nd SFR area, however, is 1 byte longer than an instruction that accesses the SFR area. 2nd SFRs can be manipulated like general-purpose registers, using operation, transfer, and bit memory manipulation instructions. The manipulable bit units, 1, 8, and 16, depend on the 2nd SFR type. Manipulation bit unit can be specified as follows.

- 1-bit manipulation

Describe as follows for the 1-bit memory manipulation instruction operand (sfr.bit)

When the bit name is defined: <Bit name>

When the bit name is not defined: <Register name>, <Bit number> or <Address>, <Bit number>

- 8-bit manipulation

Describe the symbol defined by the assembler for the 8-bit memory manipulation instruction operand (sfr). This manipulation can also be specified with an address.

- 16-bit manipulation

Describe the symbol defined by the assembler for the 16-bit memory manipulation instruction operand (sfrp). When specifying an address, describe an even address.

Section 3 Registers

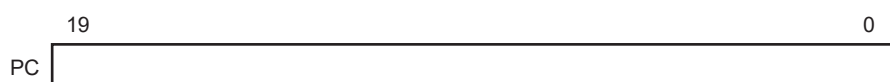
3.1 Control Registers

The control registers provide dedicated functions for program sequencing, processor state management, and stack memory control. The control registers consist of the program counter (PC), the program status word (PSW) and the stack pointer (SP).

3.1.1 Program counter (PC)

The program counter is a 20-bit register that holds the address information of the next program to be executed. In normal operation, PC is automatically incremented according to the number of bytes of the instruction to be fetched. When a branch instruction is executed, immediate data and register contents are set. Following a reset, the values of the reset vector table at addresses 0000H and 0001H are set in the 16 lower-order bits of the program counter, and the four higher-order bits are cleared to 0000.

Figure 3 - 1 Format of Program Counter

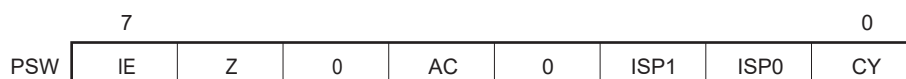


3.1.2 Program status word (PSW)

The program status word is an 8-bit register consisting of various flags that are set or reset in response to the execution of instructions. The ISP1 flag is defined as bit 2 in products that support interrupt level 4. The contents of the program status word are stored in the stack area when a vectored interrupt request is acknowledged or the PUSH PSW instruction is executed, and are restored when the RETB, RETI, or POP PSW instruction is executed. The value of this register following a reset is 06H.

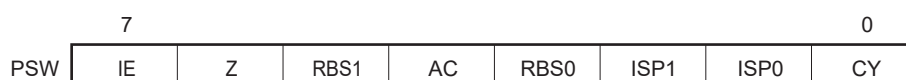
- RL78-S1 core

Figure 3 - 2 Program Status Word Configuration



- RL78-S2 core, RL78-S3 core

Figure 3 - 3 Program Status Word Configuration



1. Interrupt enable flag (IE)

The IE flag controls interrupt request acknowledgment by the CPU. An IE flag value of 0 selects the interrupt disabled (DI) state and disables all maskable interrupt requests. An IE flag value of 1 selects the interrupt enabled (EI) state, in which interrupt request acknowledgment is controlled by the in-service priority flags (ISP1 and ISP0), the interrupt mask flags for the interrupt sources, and the priority specification flags. This flag is reset to 0 upon execution of the DI instruction or interrupt request acknowledgment and is set to 1 upon execution of the EI instruction.

2. Zero flag (Z)

The zero flag is set to 1 if the result of an operation is zero; otherwise, it is reset to 0.

3. Register bank select flags (RBS0 and RBS1)

These are 2-bit flags used to select one of the four register banks. In these flags, the 2-bit information that indicates the register bank selected by SEL RBN instruction execution is stored.

Caution RBS0 and RBS1 are not mounted on the RL78-S1 core.

4. Auxiliary carry flag (AC)

This flag is set to 1 if a carry from bit 3 or a borrow at bit 3 occurs during an operation; otherwise, it is reset to 0.

5. In-service priority flags (ISP0 and ISP1)

This flag manages the priority of acknowledgeable maskable vectored interrupts. The vectored interrupt requests specified as lower than the ISP0 and ISP1 values by the priority specification flag register (PR) are disabled for acknowledgment. Actual acknowledgment for interrupt requests is controlled by the state of the interrupt enable flag (IE).

6. Carry flag (CY)

This flag indicates an overflow or underflow during the execution of an add or subtract instruction. It also holds the shifted-out value from the execution of a rotate instruction and functions as a bit accumulator in the execution of a bit operation instruction.

3.1.3 Stack pointer (SP)

The stack pointer is a 16-bit register that holds the address where the stack area starts in memory. The stack area can only be set within the internal RAM area.

Figure 3 - 4 Stack Pointer Configuration

Symbol	15	14	13	12	11	10	9	8
SP	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
	7	6	5	4	3	2	1	0
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	0

In stack addressing through the stack pointer, the SP is decremented before data are saved in the stack memory and incremented after data are restored from the stack memory. The contents of the SP following a reset are undefined. Initialize the SP to a valid address for the stack before using the stack. The SP must also be set to an even value. If an odd number is specified, the least significant bit is automatically cleared to 0. **Table 3 - 1** shows the amount of stack space used by each instruction or operation in an RL78 microcontroller.

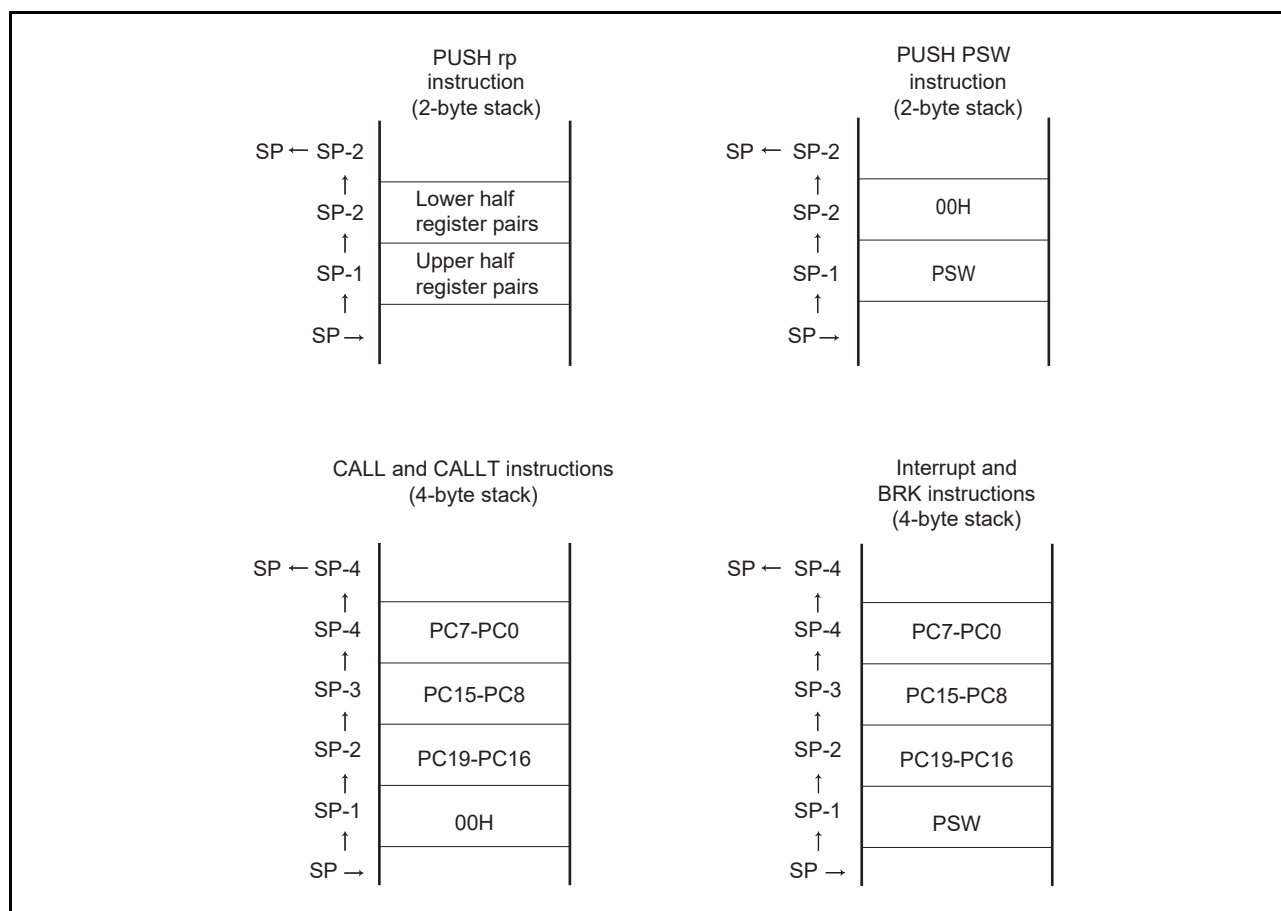
Caution The space to which the general-purpose registers are allocated cannot be used as the stack area.

Table 3 - 1 Amount of Stack Space Used by Each Instruction or Operation in an RL78 Microcontroller

Save Instruction	Restore Instruction	Stack Size
PUSH rp	POP rp	2 bytes
PUSH PSW	POP PSW	2 bytes
CALL, CALLT	RET	4 bytes
Interrupt	RETI	4 bytes
BRK	RETB	4 bytes

Figure 3 - 5 shows the data saved by various stack operations in the RL78 microcontrollers.

Figure 3 - 5 Data to Be Saved to Stack Memory



Stack pointers can be specified only within internal RAM. The target address range is from F0000H to FFFFFH; be sure not to exceed the internal RAM space. If an address outside the internal RAM space is specified, write operations to that address will be ignored and read operations will return undefined values.

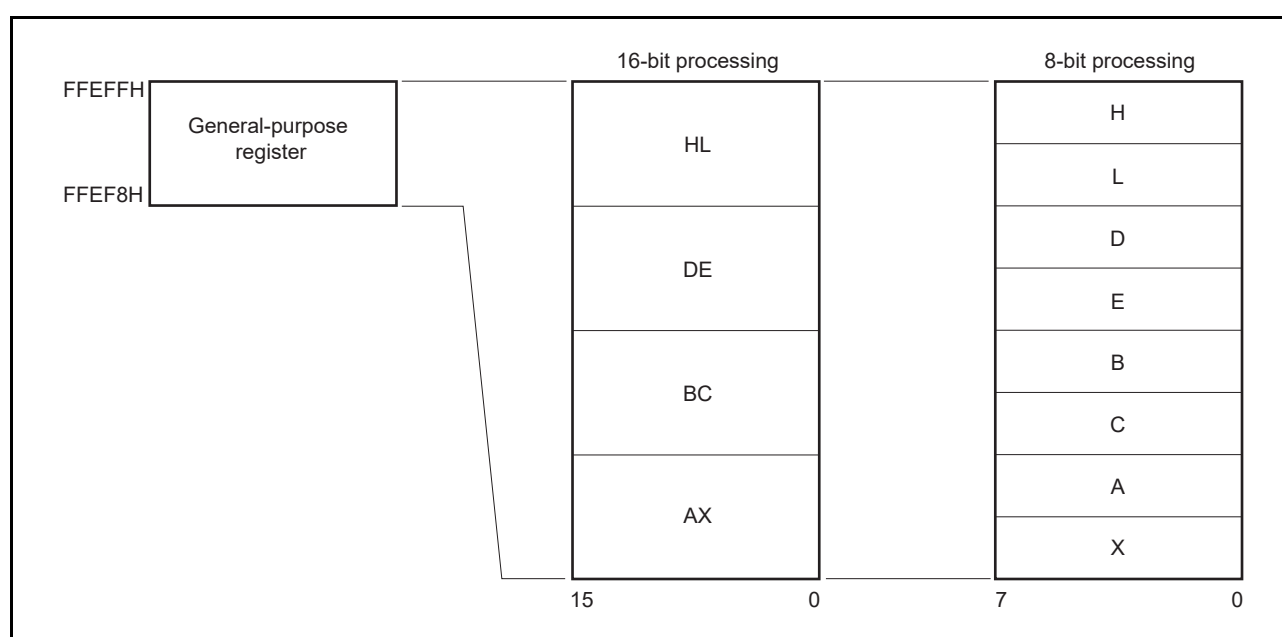
3.2 General-purpose Registers

3.2.1 General-purpose registers of RL78-S1 core

General-purpose registers are mapped at particular addresses (FFEE0H to FFEFFH) of the data memory. These registers consist of 4 banks, each bank consisting of eight 8-bit registers (X, A, C, B, E, D, L, and H). In addition that each register can be used as an 8-bit register, two 8-bit registers in pairs can be used as a 16-bit register (AX, BC, DE, and HL).

Caution Use of the general-purpose register space (FFEE8H to FFEFFH) as the instruction fetch area or stack area is prohibited.

Figure 3 - 6 Configuration of General-purpose Registers of RL78-S1 Core



3.2.2 General-purpose registers of RL78-S2 core and RL78-S3 core

General-purpose registers are mapped at particular addresses (FFEE0H to FFEFFH) of the data memory. The general-purpose registers consists of 4 banks, each bank consisting of eight 8-bit registers (X, A, C, B, E, D, L, and H). Each register can be used as an 8-bit register, and two 8-bit registers can also be used in a pair as a 16-bit register (AX, BC, DE, and HL). Register banks to be used for instruction execution are set by the CPU control instruction (SEL RBn). Because of the 4-register bank configuration, an efficient program can be created by switching between a register for normal processing and a register for interrupt processing for each bank.

Caution The space to which the general-purpose registers are allocated (FFEE0H to FFEFFH) cannot be used for instruction fetching or the stack.

Figure 3 - 7 Configuration of General-purpose Registers of RL78-S2 Core and RL78-S3 Core

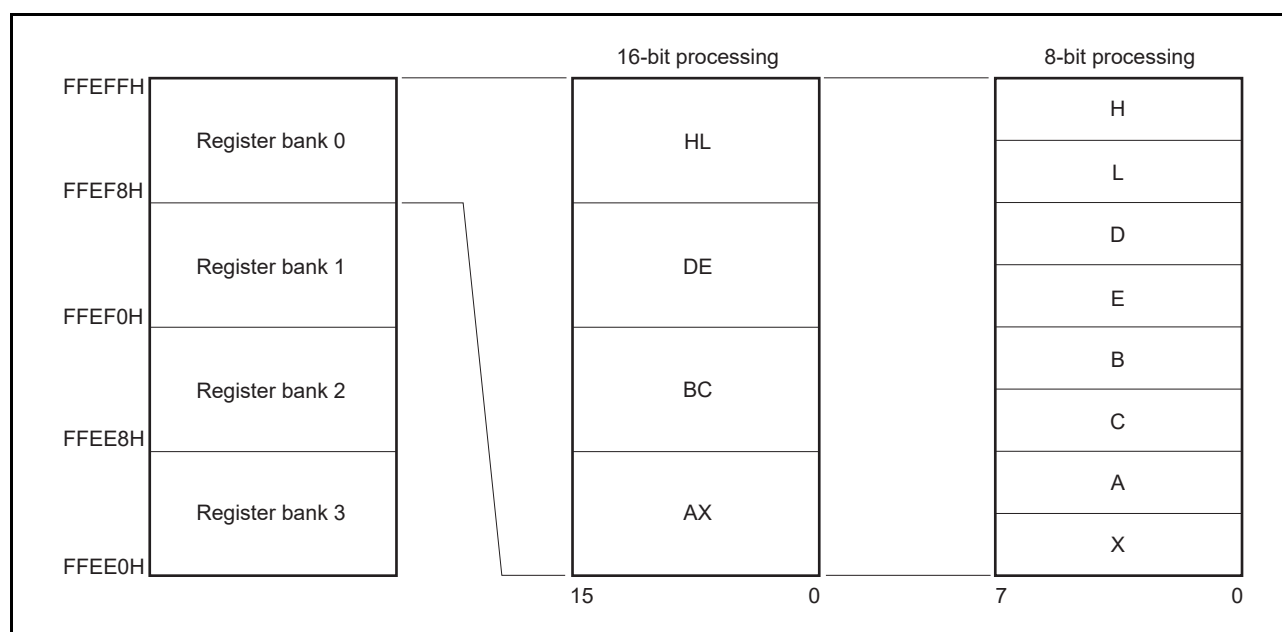


Table 3 - 2 List of General-purpose Registers

Bank Name	Register				Absolute Address
	Functional Name		Absolute Name		
	16-bit Processing	8-bit Processing	16-bit Processing	8-bit Processing	
BANK0	HL	H	RP3	R7	FFEFFH
		L		R6	FFEFEH
	DE	D	RP2	R5	FFEFDH
		E		R4	FFEFCH
	BC	B	RP1	R3	FFEFBH
		C		R2	FFEFAH
	AX	A	RP0	R1	FFEF9H
		X		R0	FFEF8H
BANK1 ^{Note}	HL	H	RP3	R7	FFEF7H
		L		R6	FFEF6H
	DE	D	RP2	R5	FFEF5H
		E		R4	FFEF4H
	BC	B	RP1	R3	FFEF3H
		C		R2	FFEF2H
	AX	A	RP0	R1	FFEF1H
		X		R0	FFEF0H
BANK2 ^{Note}	HL	H	RP3	R7	FFEEFH
		L		R6	FFEEEH
	DE	D	RP2	R5	FFEEDH
		E		R4	FFEECH
	BC	B	RP1	R3	FFEEBH
		C		R2	FFEEAH
	AX	A	RP0	R1	FFEE9H
		X		R0	FFEE8H
BANK3 ^{Note}	HL	H	RP3	R7	FFEE7H
		L		R6	FFEE6H
	DE	D	RP2	R5	FFEE5H
		E		R4	FFEE4H
	BC	B	RP1	R3	FFEE3H
		C		R2	FFEE2H
	AX	A	RP0	R1	FFEE1H
		X		R0	FFEE0H

Note Not mounted on the RL78-S1 core.

3.3 ES and CS Registers

The ES register is used to specify the higher-order bits of addresses for data access. The CS register is used to specify the higher-order bits of addresses for branch instructions that use register direct addressing. For details on the use of these registers, see **Section 4 Addressing**. The values of the ES and CS registers following a reset are 0FH and 00H, respectively.

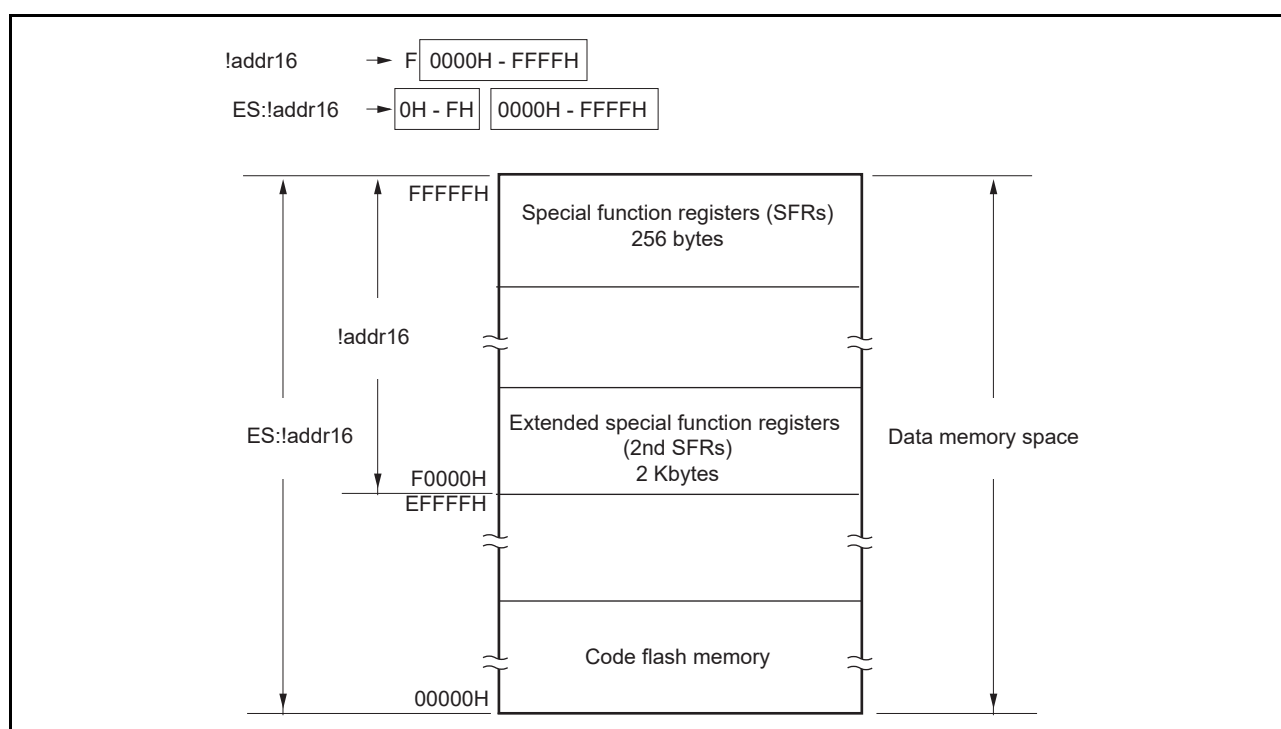
Figure 3 - 8 Configuration of ES and CS Registers

	7	6	5	4	3	2	1	0
ES	0	0	0	0	ES3	ES2	ES1	ES0

	7	6	5	4	3	2	1	0
CS	0	0	0	0	CS3	CS2	CS1	CS0

The 64-Kbyte data area from F0000H to FFFFFH is accessible by using 16-bit addressing. Using ES: as the prefix for an address extends the accessible data area to the 1-Mbyte address space from 00000H to FFFFFH.

Figure 3 - 9 Extension of the Accessible Data Area



3.4 Special Function Registers (SFRs)

Table 3 - 3 describes fixed-address SFRs in the RL78 microcontrollers.

Table 3 - 3 List of Fixed SFRs

Address	Register Name
FFFF8H	SPL
FFFF9H	SPH
FFFFAH	PSW
FFFFBH	Reserved
FFFFCH	CS
FFFFDH	ES
FFFFEH	PMC
FFFFFH	MEM

3.4.1 Processor mode control register (PMC)

The PMC is an 8-bit register that is used to control the processor modes. For details, see **2.2 Internal Program Memory Space**. The value of this register following a reset is 00H.

1. RL78-S1 core

Figure 3 - 10 Configuration of Processor Mode Control Register (PMC) of RL78-S1 Core

Address: FFFFEH
 After reset: 00H
 R/W: R/W

Symbol	7	6	5	4	3	2	1	<0>
PMC	0	0	0	0	0	0	0	MAA

MAA	Selection of flash memory space mirroring to area from F8000H to FDEFFH ^{Note}
0	00000H to 05EFFH is mirrored at F8000H to FDEFFH
1	Setting prohibited

Note The range from F8000H to FDEFFH is allocated to the SFR and RAM areas as well as the mirrored flash memory area. Wherever the SFR and RAM areas overlap with the mirrored flash memory area, the SFR and RAM areas take priority.

Caution The PMC should remain at its initial value and should not be rewritten. To maintain compatibility with the RL78-S2 and RL78-S3 cores, however, only 00H can be written to the PMC.

2. RL78-S2 core, RL78-S3 core

Figure 3 - 11 Configuration of Processor Mode Control Register (PMC) of RL78-S2 Core and RL78-S3 Core

Address: FFFFEH
After reset: 00H
R/W: R/W

Symbol	7	6	5	4	3	2	1	<0>
PMC	0	0	0	0	0	0	0	MAA

<R>	MAA	Selection of flash memory space mirroring to area from F0000H to FFFFFH ^{Note}
	0	00000H to 0FFFFH is mirrored at F0000H to FFFFFH
	1	10000H to 1FFFFH is mirrored at F0000H to FFFFFH

Note The range from F0000H to FFFFFH is allocated to the SFR and RAM areas as well as the mirrored flash memory area. Wherever the SFR and RAM areas overlap with the mirrored flash memory area, the SFR and RAM areas take priority.

Caution After setting PMC, wait for at least one instruction and access the mirror area.

Section 4 Addressing

Addressing is classified into two categories: program addressing and data addressing. This section describes the addressing modes for each category.

4.1 Program Addressing

4.1.1 Relative addressing

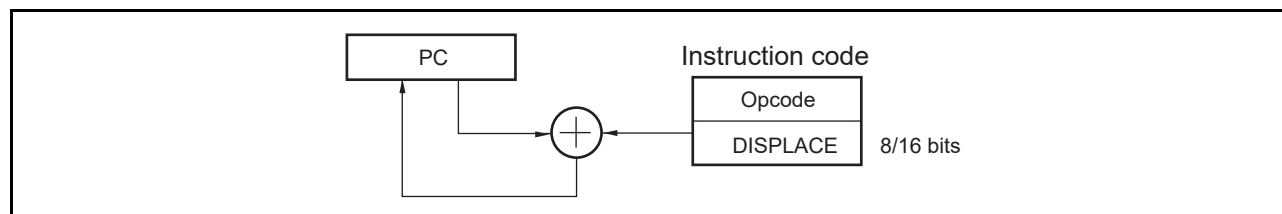
[Function]

In relative addressing, a displacement value^{Note 1} included as the operand for the instruction is added to the program counter (PC) value^{Note 2}. The resulting value is stored in the program counter and used as the branch destination address. Relative addressing is only applied to branch instructions.

Note 1. Signed two's complement data: -128 to +127 or -32768 to +32767

Note 2. The program counter value used in the calculation is the address where the next instruction starts.

Figure 4 - 1 Outline of Relative Addressing



4.1.2 Immediate addressing

[Function]

With immediate addressing, the CPU loads the program counter with the immediate data in the instruction word as the branch destination address. For immediate addressing, CALL !!addr20 or BR !!addr20 is used to specify a 20-bit address and CALL !addr16 or BR !addr16 is used to specify a 16-bit address. For a 16-bit address, the CPU sets the four higher-order bits to 0000.

Figure 4 - 2 Example of CALL !!addr20 and BR !!addr20

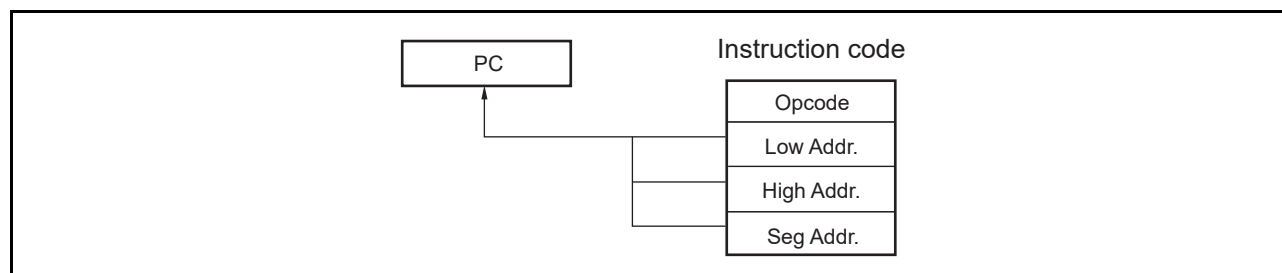
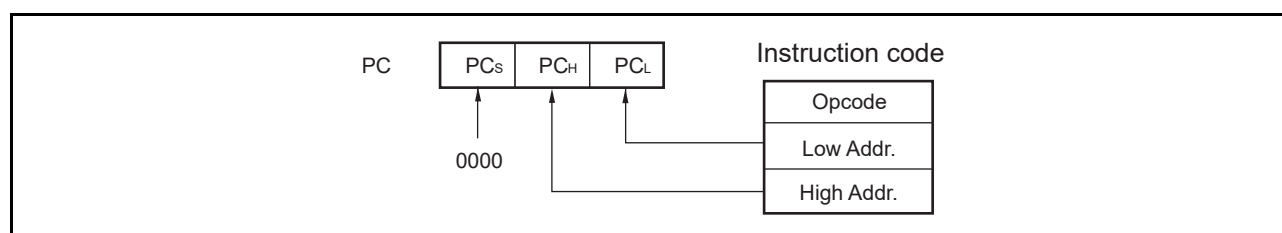


Figure 4 - 3 Example of CALL !addr16 and BR !addr16

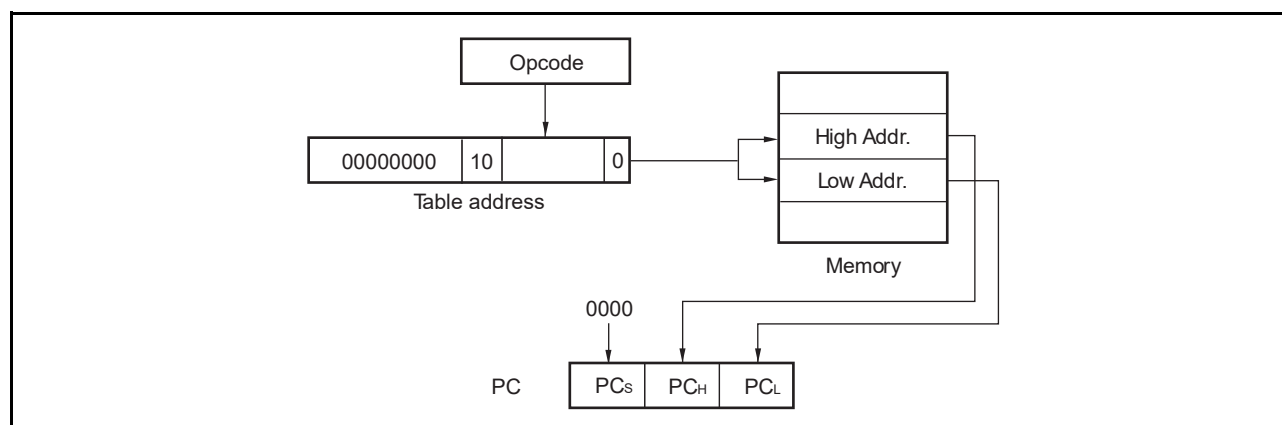


4.1.3 Table indirect addressing

[Function]

With table indirect addressing, the CPU loads the five bits of immediate data in the instruction word to specify a table address in the CALLT table area (0080H to 00BFH). The byte at the specified address in the table and the byte at the next address are loaded into the program counter (PC) as a 16-bit branch destination address. Table indirect addressing is only used with CALLT instructions. In RL78 microcontrollers, this only enables branching to the 64-Kbyte space from 00000H to 0FFFFH.

Figure 4 - 4 Outline of Table Indirect Addressing



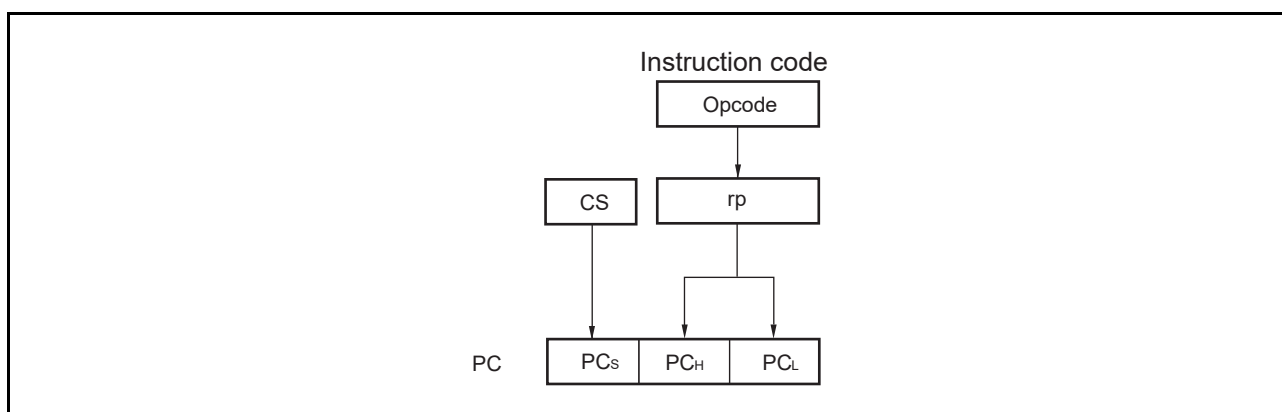
<R> 4.1.4 Register indirect addressing

[Function]

With register indirect addressing, the CPU loads the program counter with the contents of the CS register and a general-purpose register pair (AX, BC, DE, or HL) in the current register bank as a 20-bit branch destination address. Register

<R> indirect addressing is only applied to the CALL AX, BC, DE, HL, and BR AX instructions.

<R> Figure 4 - 5 Outline of Register Indirect Addressing



4.2 Data Addressing

4.2.1 Implied addressing

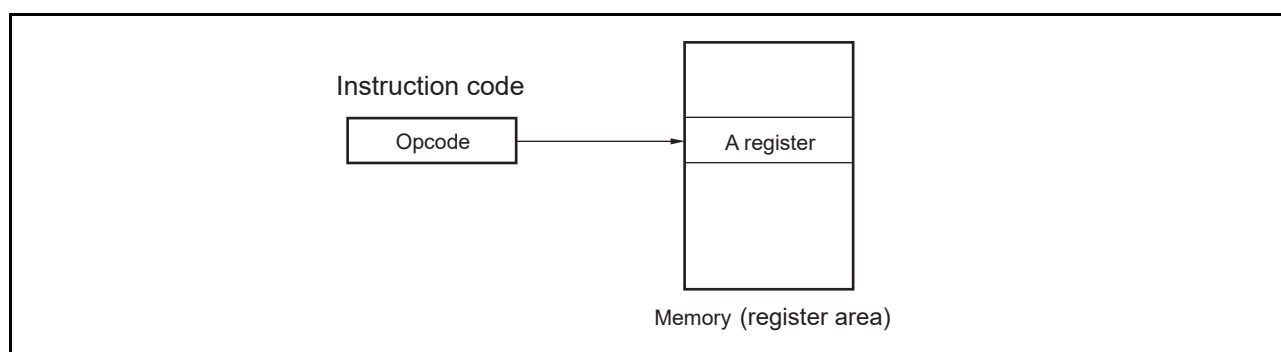
[Function]

Instructions for accessing registers (such as accumulators) that have special functions are directly specified with the instruction word, without using any register specification field in the instruction word.

[Operand format]

Implied addressing is only applied to MULU X.

Figure 4 - 6 Outline of Implied Addressing



4.2.2 Register addressing

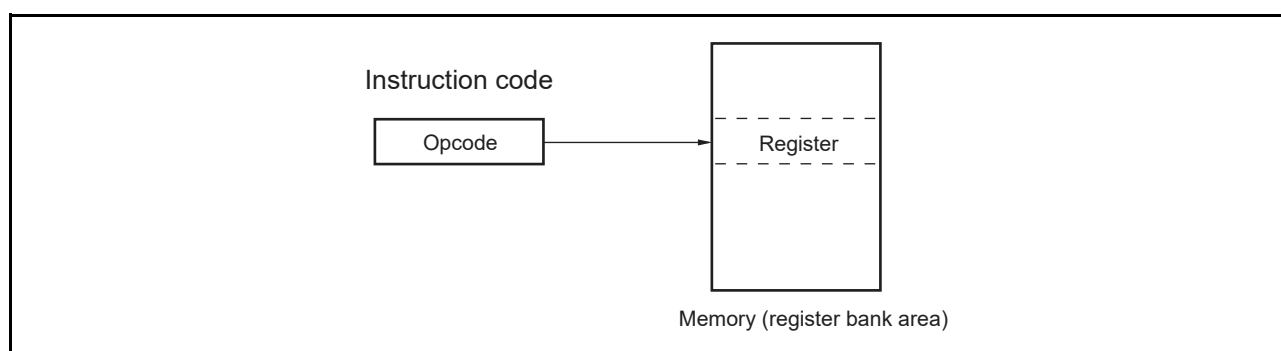
[Function]

Register addressing accesses a general-purpose register as an operand. The instruction word of 3-bit long is used to select an 8-bit register and the instruction word of 2-bit long is used to select a 16-bit register.

[Operand format]

Identifier	Description
r	X, A, C, B, E, D, L, H
rp	AX, BC, DE, HL

Figure 4 - 7 Outline of Register Addressing



4.2.3 Direct addressing

[Function]

Direct addressing uses immediate data in the instruction word as an operand address to directly specify the target address.

[Operand format]

Identifier	Description
!addr16	Label or 16-bit immediate data (only the space from F0000H to FFFFFH is specifiable)
ES: !addr16	Label or 16-bit immediate data (the four higher-order bits are specified by the ES register)

Figure 4 - 8 Example of !addr16

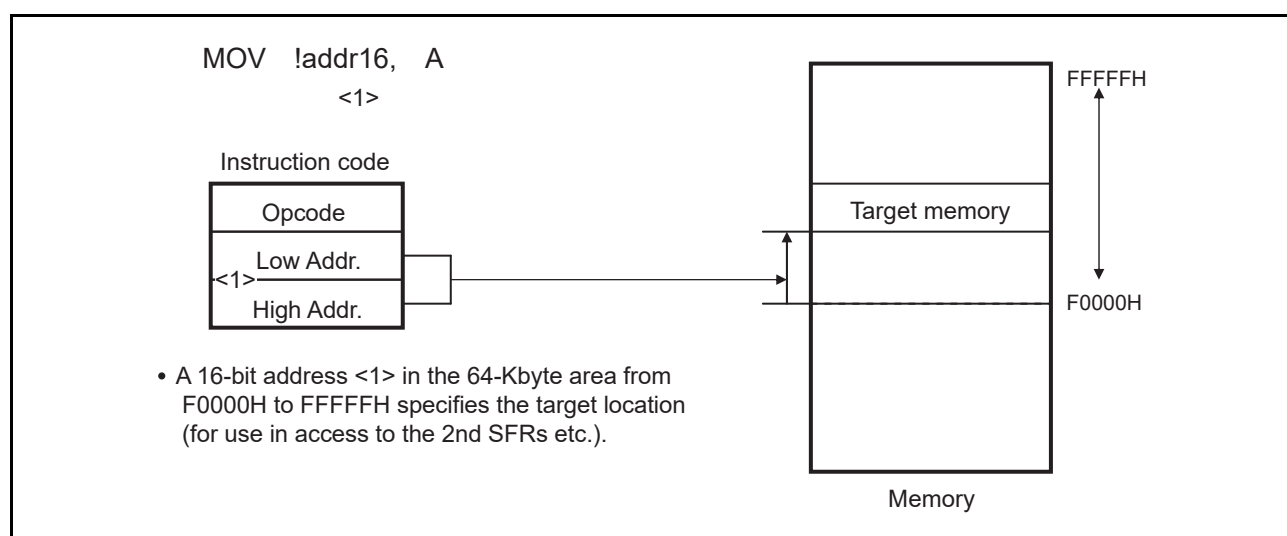
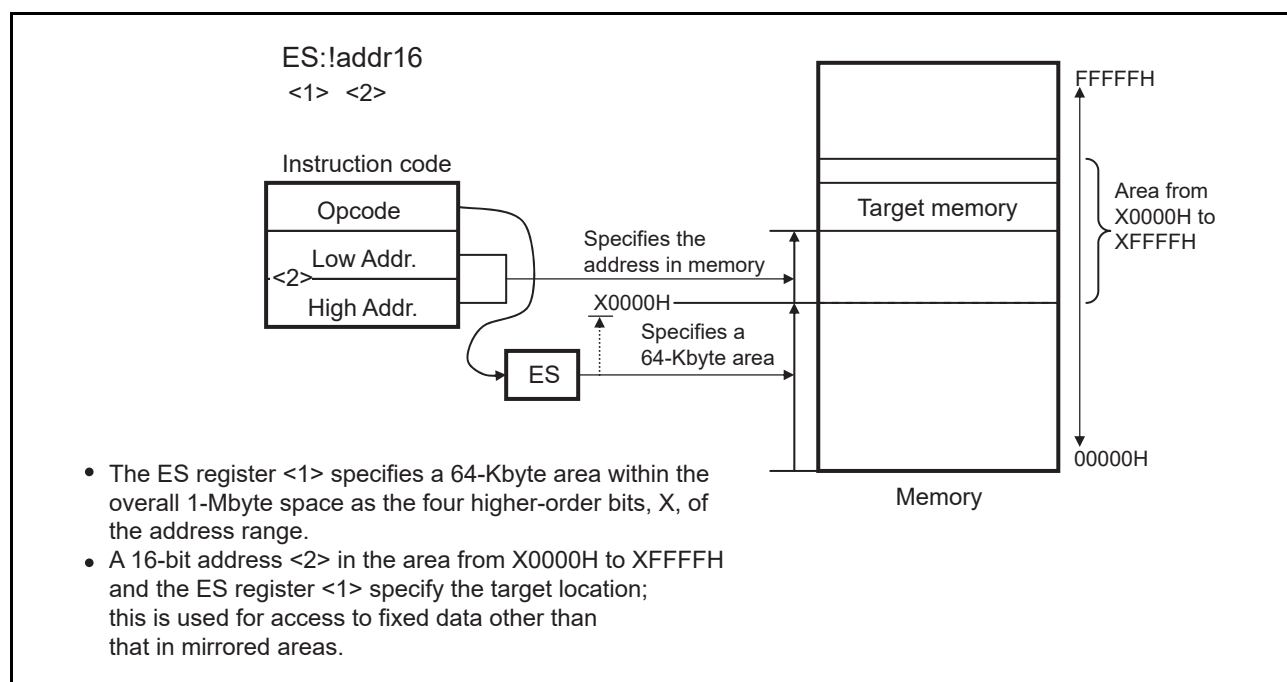


Figure 4 - 9 Example of ES:!addr16



4.2.4 Short direct addressing

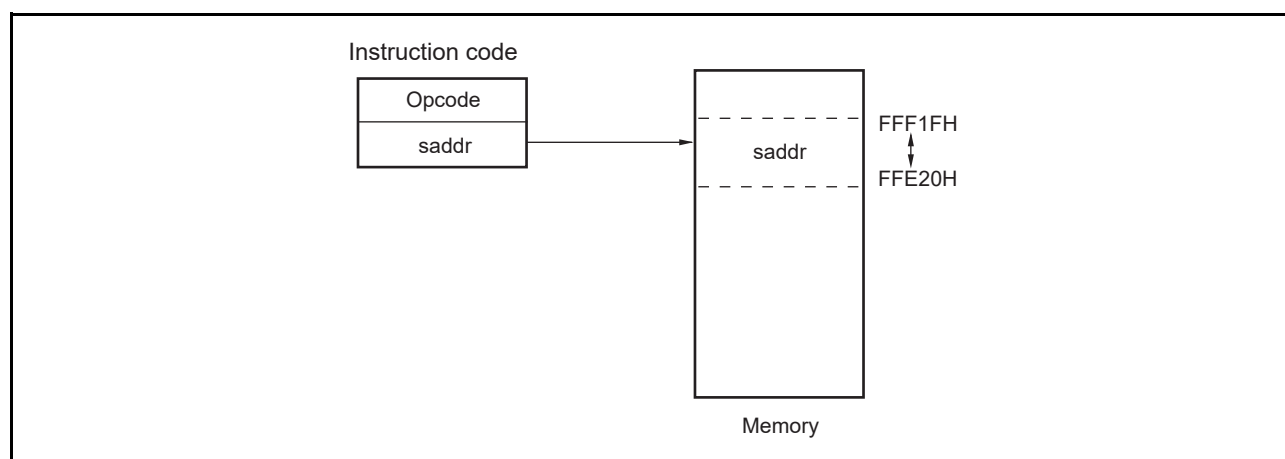
[Function]

Short direct addressing directly specifies the target addresses using 8-bit data in the instruction word. Short direct addressing is only applied to the space from FFE20H to FFF1FH.

[Operand format]

Identifier	Description
SADDR	Label, FFE20H to FFF1FH immediate data or 0FE20H to 0FF1FH immediate data (only the space from FFE20H to FFF1FH is specifiable)
SADDRP	Label, FFE20H to FFF1FH immediate data or 0FE20H to 0FF1FH immediate data (even address only) (only the space from FFE20H to FFF1FH is specifiable)

Figure 4 - 10 Outline of Short Direct Addressing



Remark SADDR and SADDRP are used to describe the values of addresses FE20H to FF1FH with 16-bit immediate data (the four higher-order bits of the actual address are omitted), and the values of addresses FFE20H to FFF1FH with 20-bit immediate data. Regardless of whether SADDR or SADDRP is used, addresses within the space from FFE20H to FFF1FH are specified for the memory.

4.2.5 SFR addressing

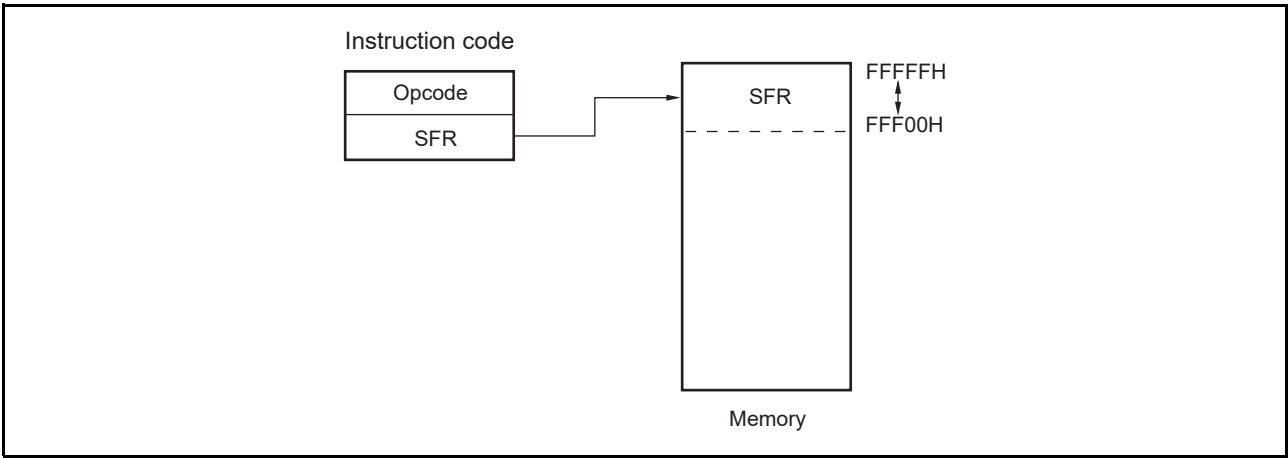
[Function]

SFR addressing directly specifies the target SFR addresses using 8-bit data in the instruction word. SFR addressing is only applied to the space from FFF00H to FFFFFH.

[Operand format]

Identifier	Description
SFR	SFR name
SFRP	16-bit-manipulatable SFR name (even address)

Figure 4 - 11 Outline of SFR Addressing



4.2.6 Register indirect addressing

[Function]

Register indirect addressing directly specifies the target addresses using the contents of the register pair specified with the instruction word as an operand address.

[Operand format]

Identifier	Description
—	[DE], [HL] (only the space from F0000H to FFFFFH is specifiable)
—	ES:[DE], ES:[HL] (the four higher-order bits are specified by the ES register)

Figure 4 - 12 Example of [DE] and [HL]

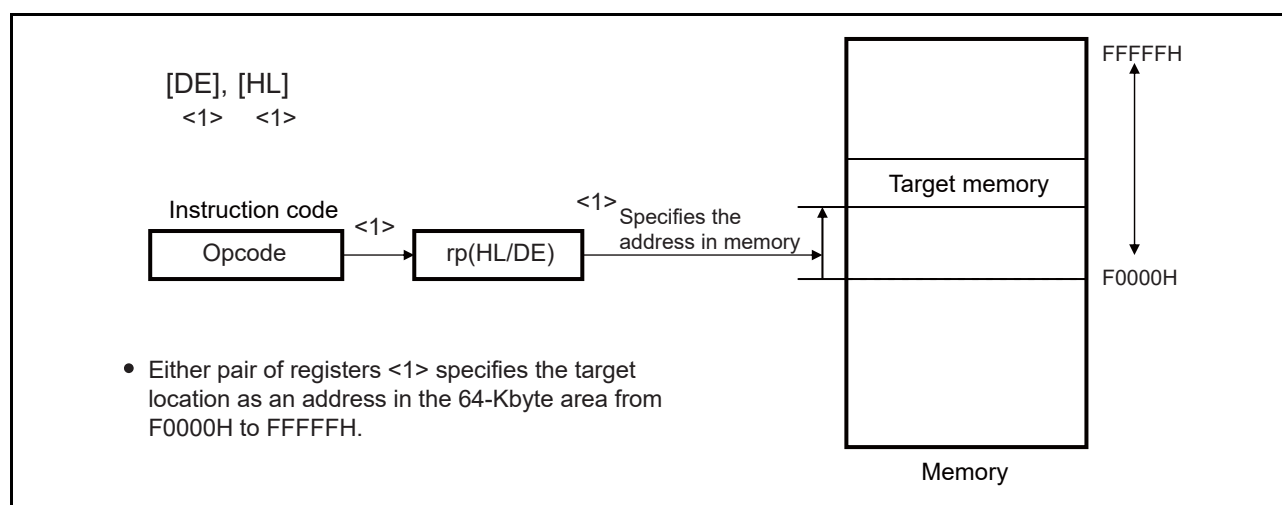
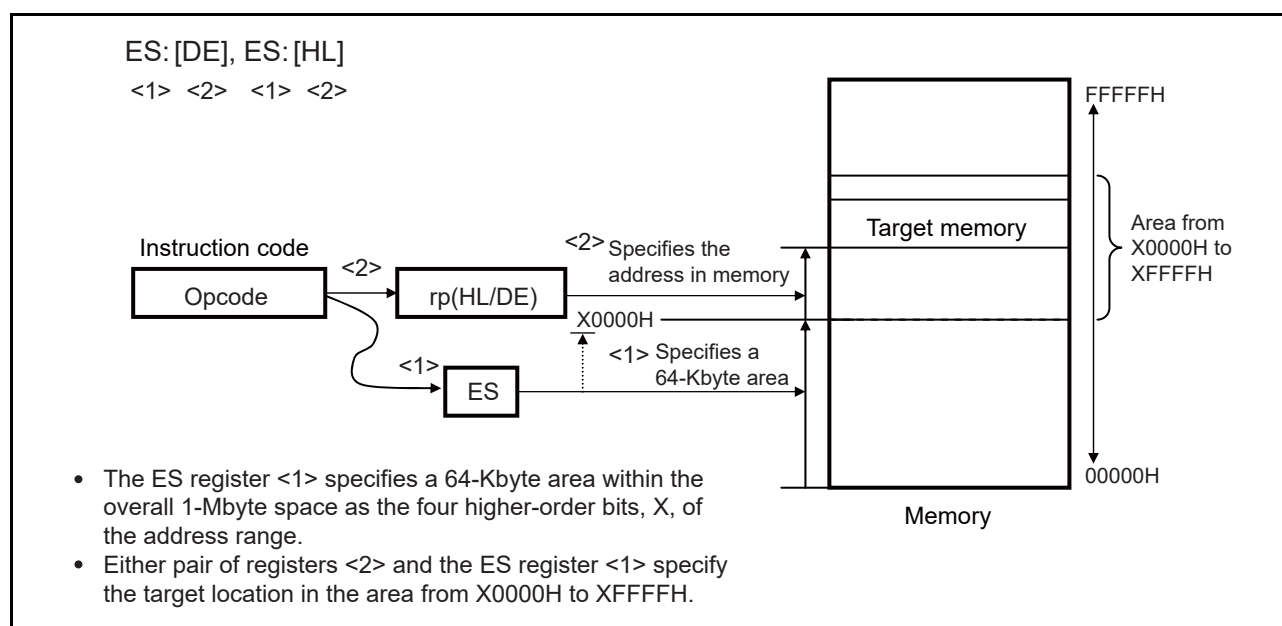


Figure 4 - 13 Example of ES:[DE] and ES:[HL]



4.2.7 Based addressing

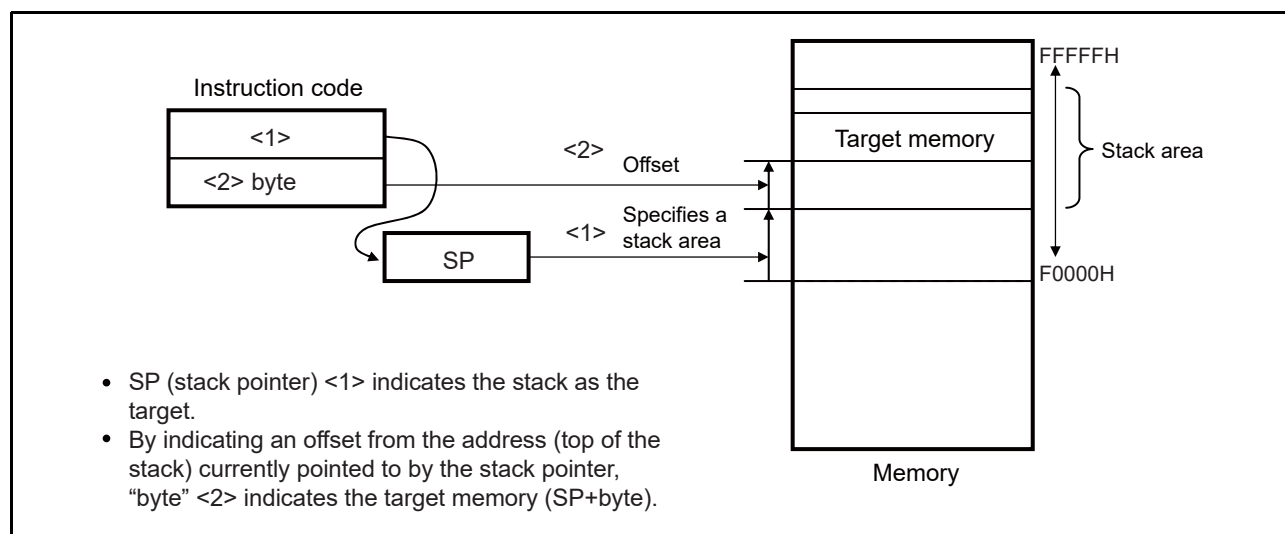
[Function]

Based addressing uses the contents of a register pair specified with the instruction word or 16-bit immediate data as a base address, and 8-bit immediate data or 16-bit immediate data as offset data. The sum of these values is used to specify the target address.

[Operand format]

Identifier	Description
—	[HL+byte], [DE+byte], [SP+byte] (only the space from F0000H to FFFFFH is specifiable)
—	word[B], word[C] (only the space from F0000H to FFFFFH is specifiable)
—	word[BC] (only the space from F0000H to FFFFFH is specifiable)
—	ES:[HL+byte], ES:[DE+byte] (the four higher-order bits are specified by the ES register)
—	ES:word[B], ES:word[C] (the four higher-order bits are specified by the ES register)
—	ES:word[BC] (the four higher-order bits are specified by the ES register)

Figure 4 - 14 Example of [SP+byte]



Caution For [HL+byte], [DE+byte], word[B], word[C], and word[BC], an added value must not exceed FFFFH. For ES:[HL+byte], ES:[DE+byte], ES:word[B], ES:word[C], and ES:word[BC], an added value must not exceed FFFFFH. For [SP+byte], an SP value must be within RAM space and the added value of SP+byte must be FFEDFH or less in RAM space.

Figure 4 - 15 Example of [HL+byte] and [DE+byte]

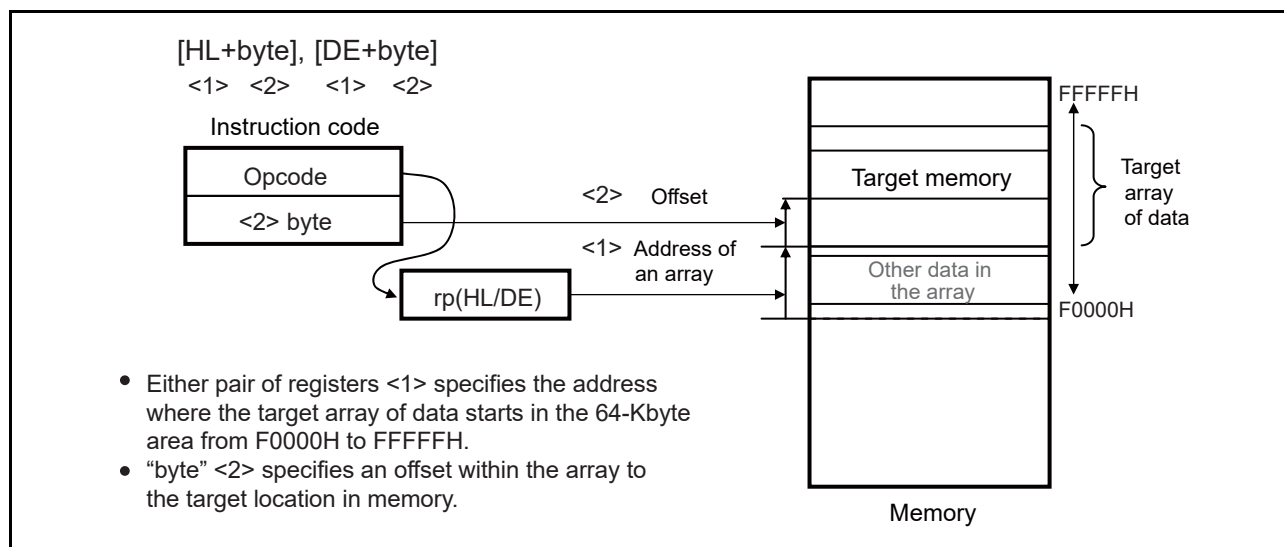


Figure 4 - 16 Example of word[B] and word[C]

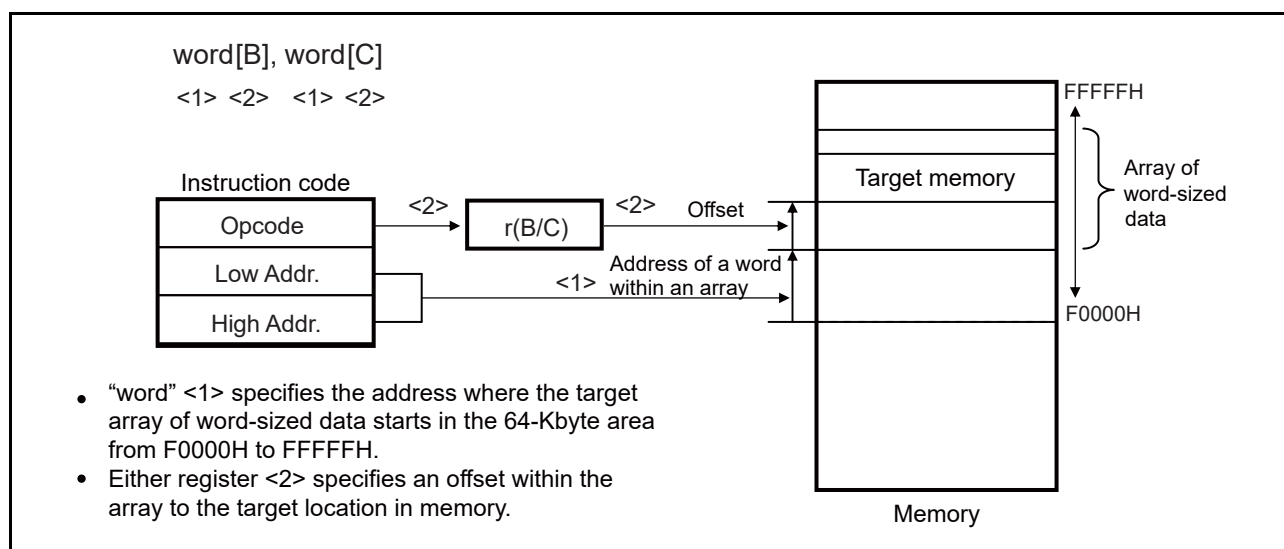


Figure 4 - 17 Example of word[BC]

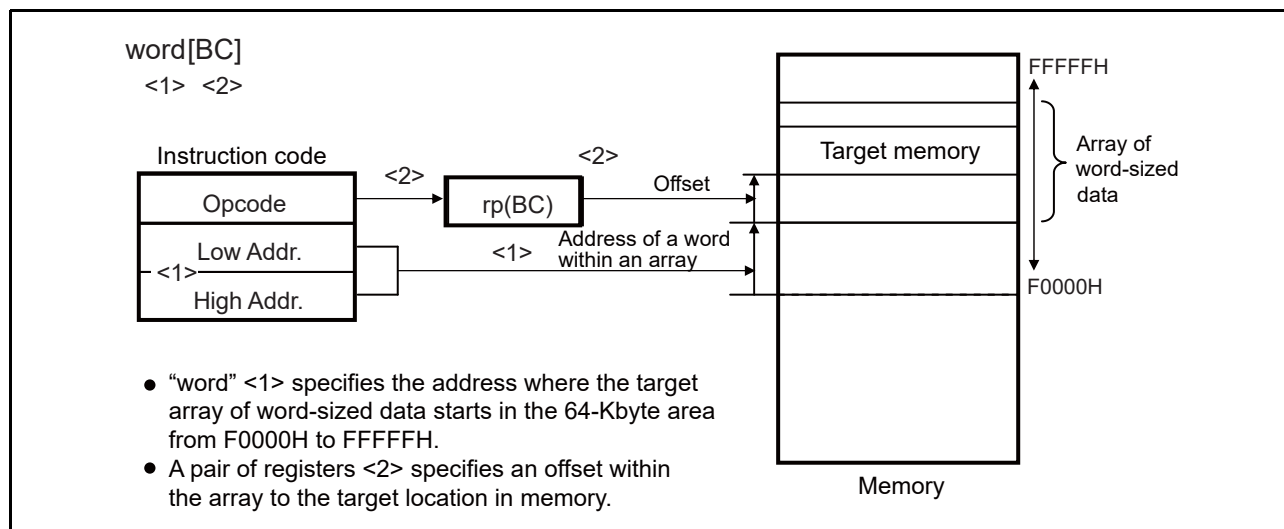


Figure 4 - 18 Example of ES:[HL+byte] and ES:[DE+byte]

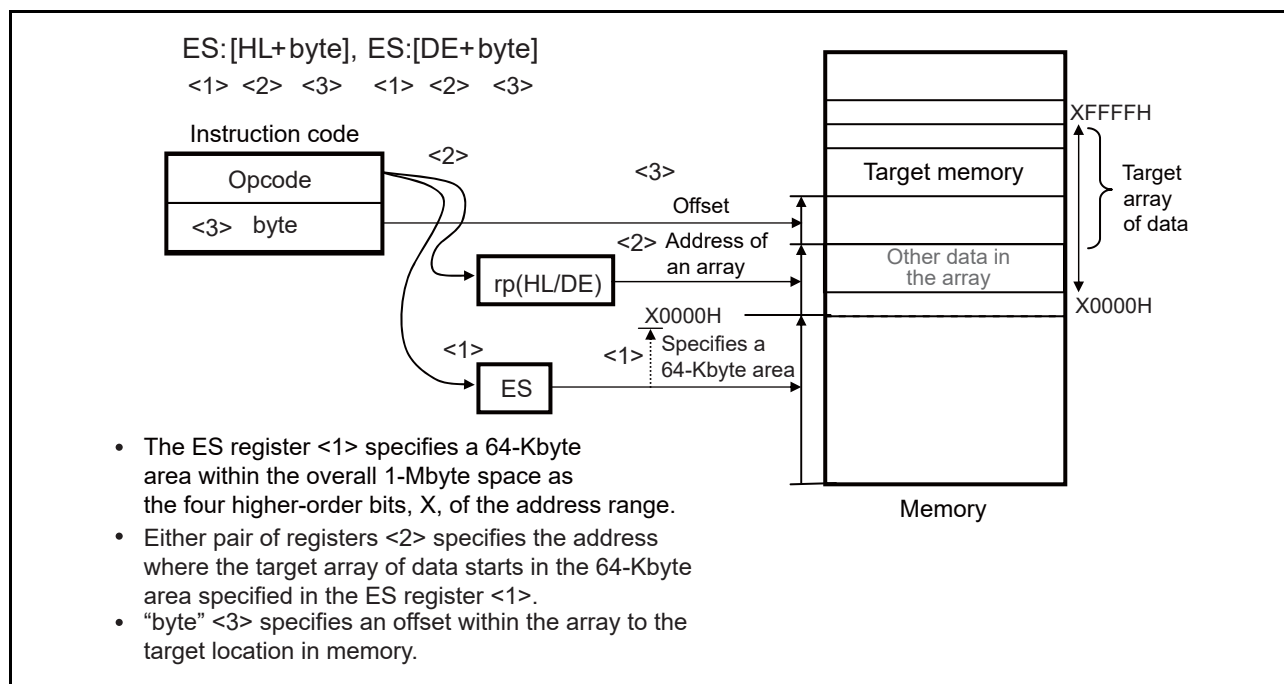


Figure 4 - 19 Example of ES:word[B] and ES:word[C]

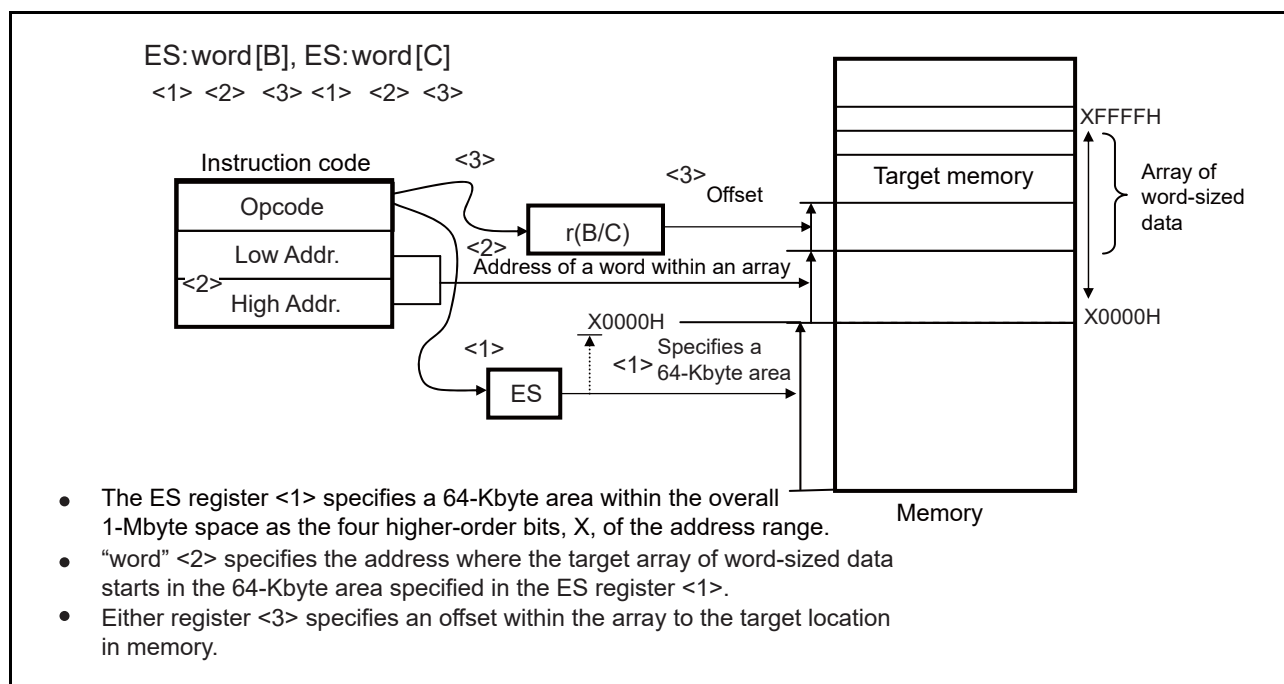
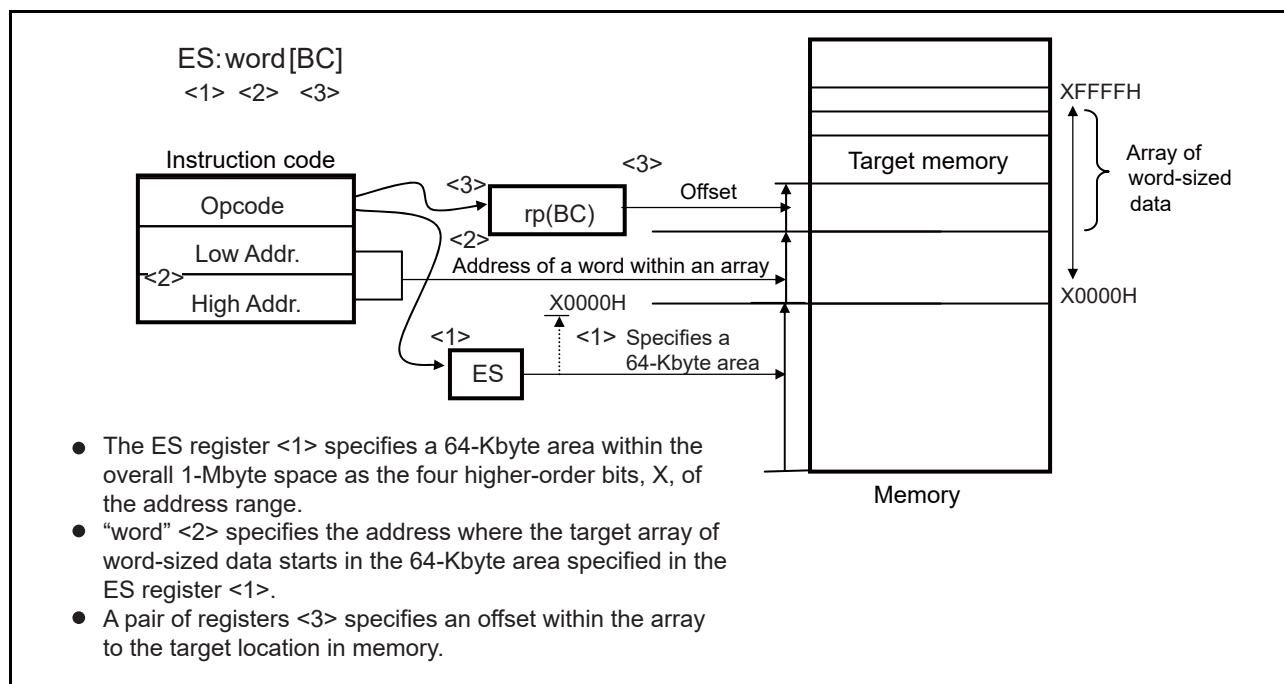


Figure 4 - 20 Example of ES:word[BC]



4.2.8 Based indexed addressing

[Function]

With based indexed addressing, the target address is specified by adding the offset address to the base address. The content of a register pair specified in the instruction word is used as the base address, and that of the B register or C register specified in the instruction word is used as the offset address.

[Operand format]

Identifier	Description
—	[HL+B], [HL+C] (only the space from F0000H to FFFFFH is specifiable)
—	ES:[HL+B], ES:[HL+C] (the four higher-order bits are specified by the ES register)

Figure 4 - 21 Example of [HL+B] and [HL+C]

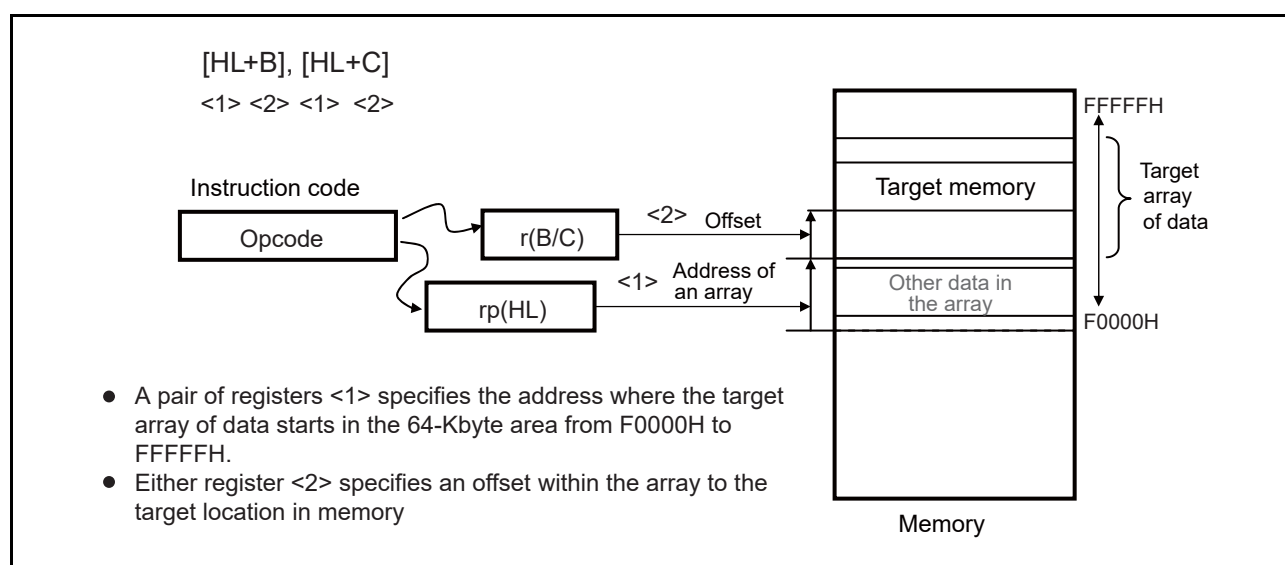
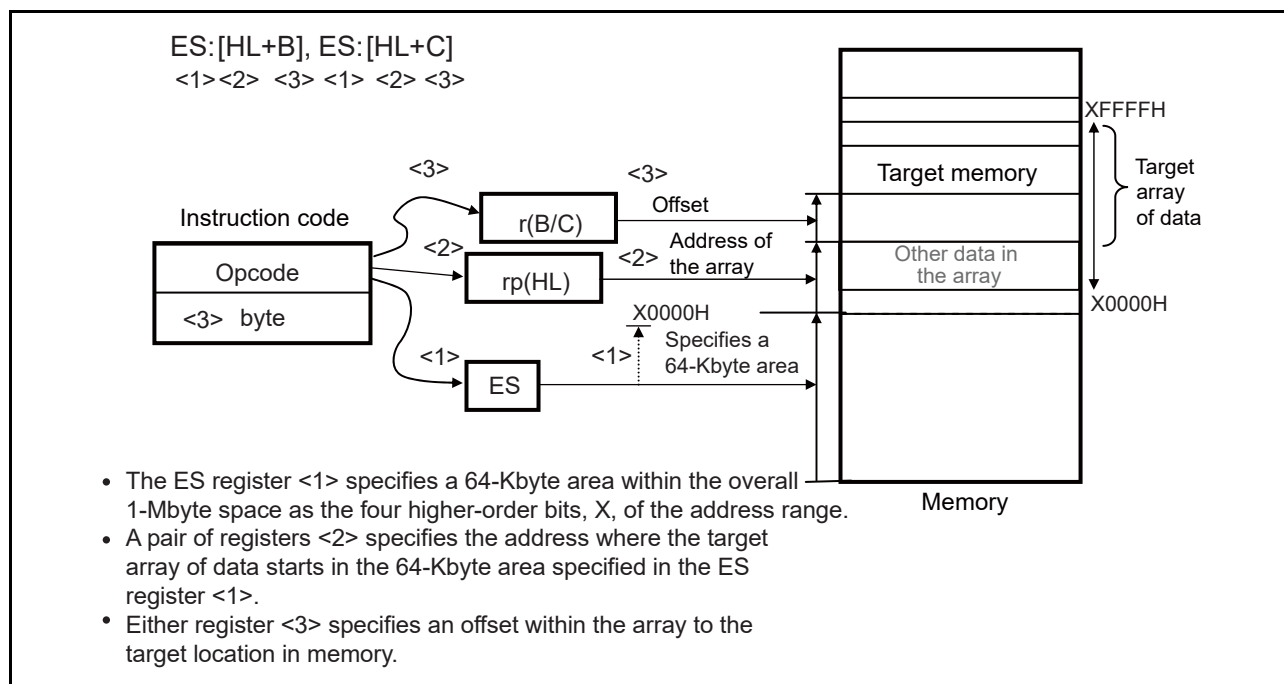


Figure 4 - 22 Example of ES:[HL+B] and ES:[HL+C]



Caution For [HL+B] and [HL+C], an added value must not exceed FFFFH.
 For ES:[HL+B] and ES:[HL+C], an added value must not exceed FFFFH.

4.2.9 Stack addressing

[Function]

The stack area is indirectly addressed with the stack pointer (SP) contents. This addressing is automatically employed when the PUSH, POP, subroutine call, and return instructions are executed or the register is saved or restored upon generation of an interrupt request. Stack addressing is only applied to the internal RAM area.

[Operand format]

Identifier	Description
—	PUSH PSW AX/BC/DE/HL POP PSW AX/BC/DE/HL CALL/CALLT RET BRK RETB (Interrupt request generated) RETI

Figures 4 - 23 to 4 - 28 show data to be saved or restored by each stack addressing.

Figure 4 - 23 Example of PUSH rp

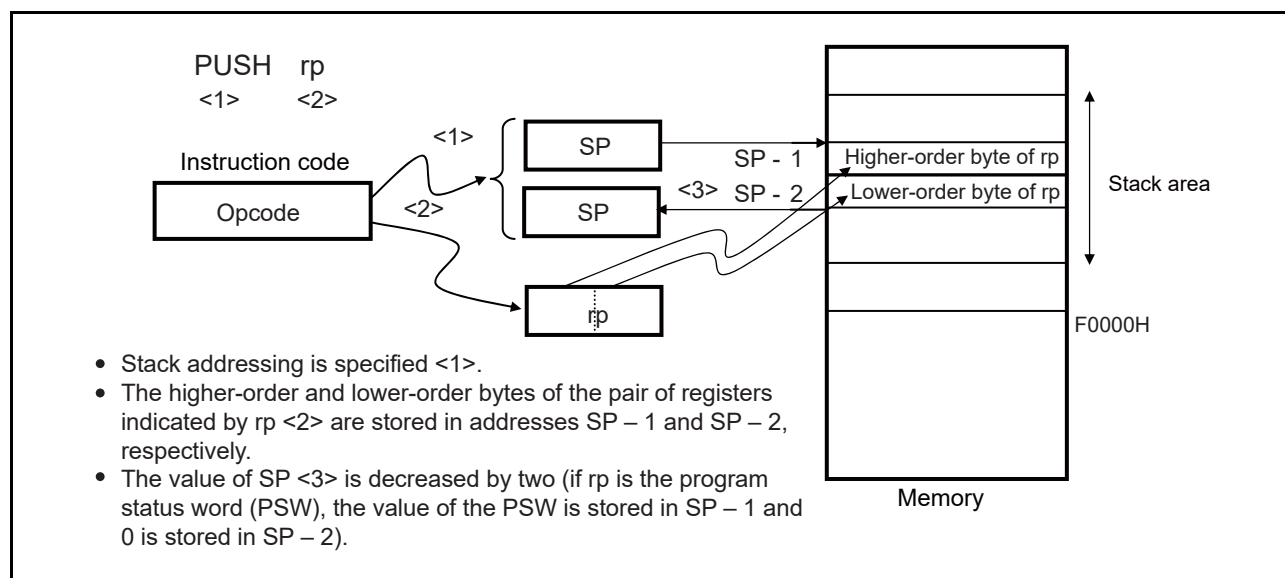


Figure 4 - 24 Example of POP

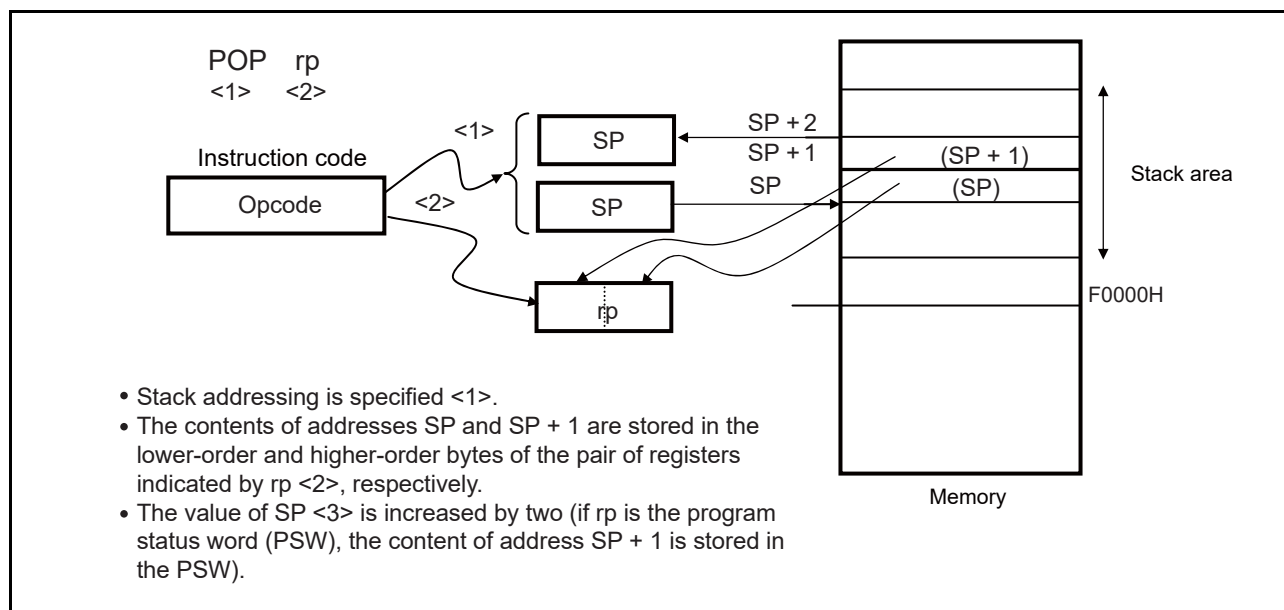


Figure 4 - 25 Example of CALL and CALLT

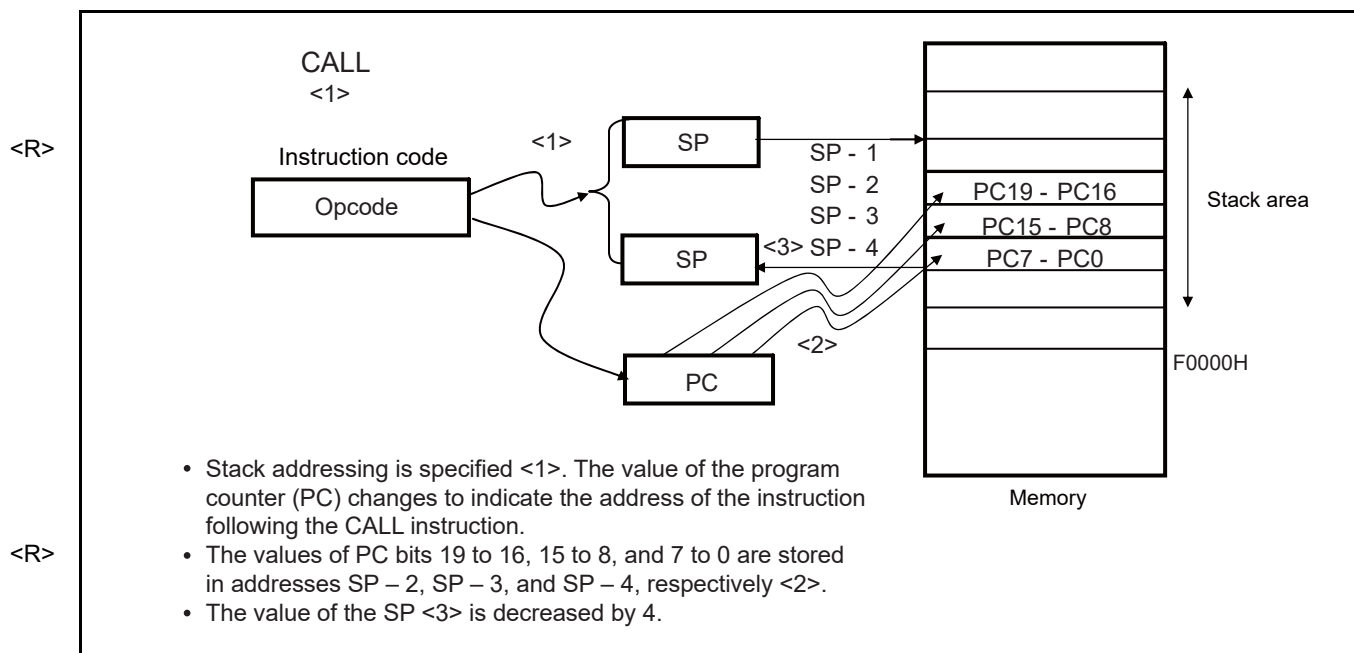


Figure 4 - 26 Example of RET

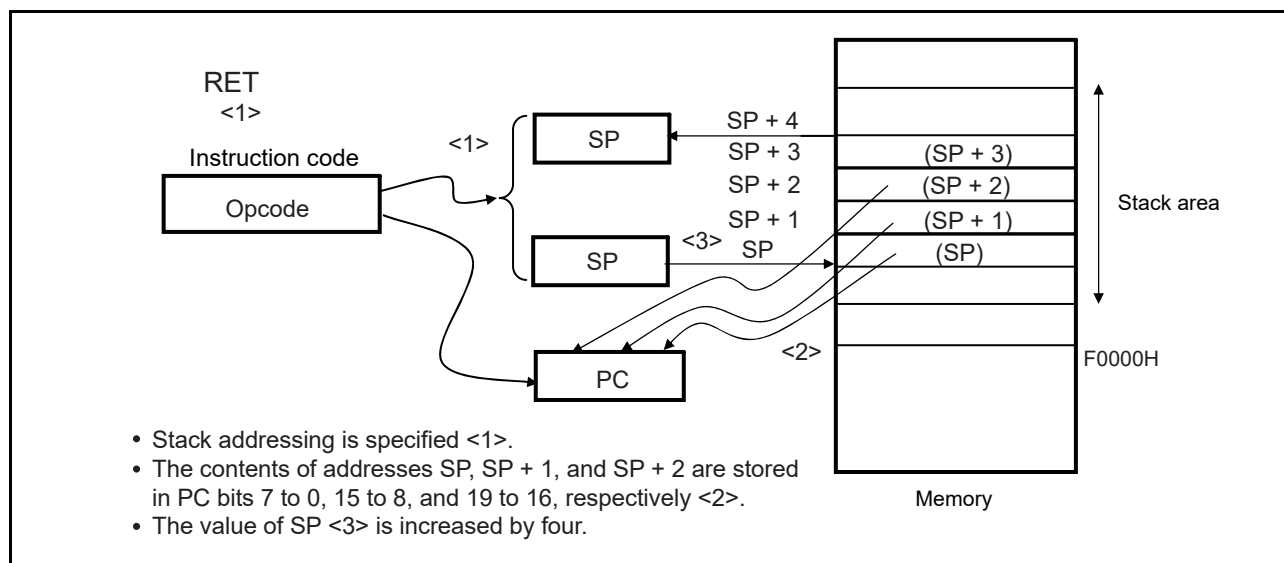


Figure 4 - 27 Example of Interrupt and BRK

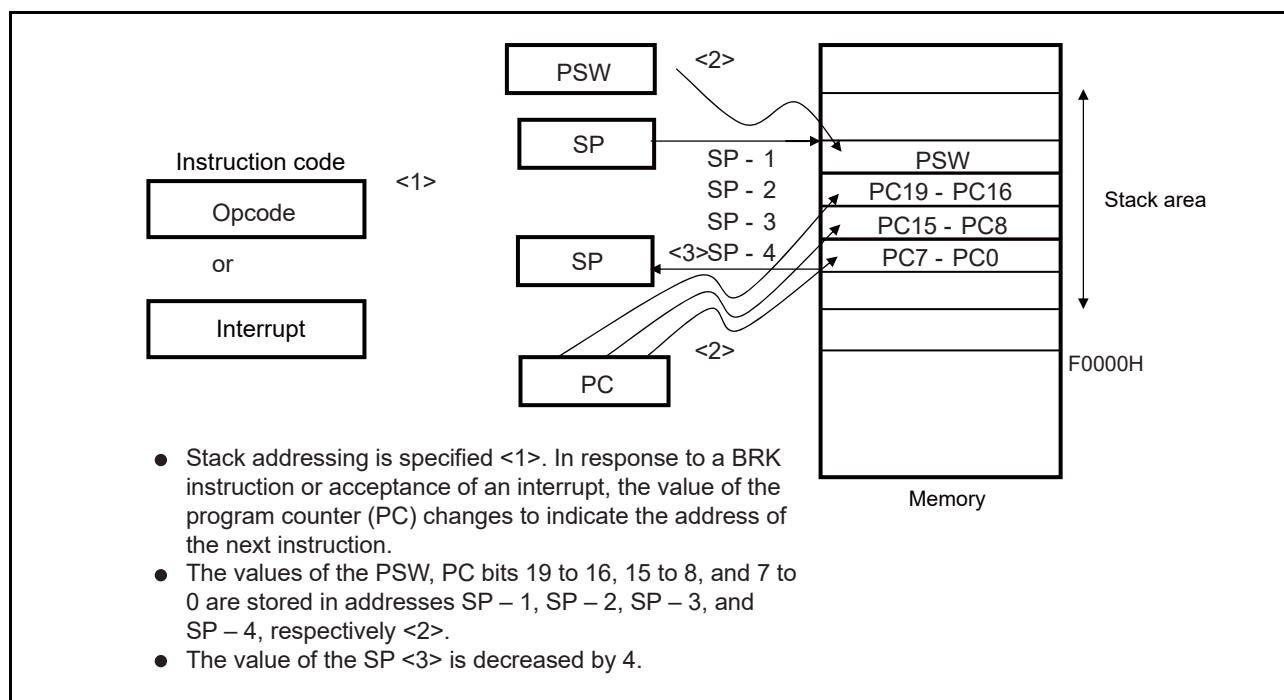
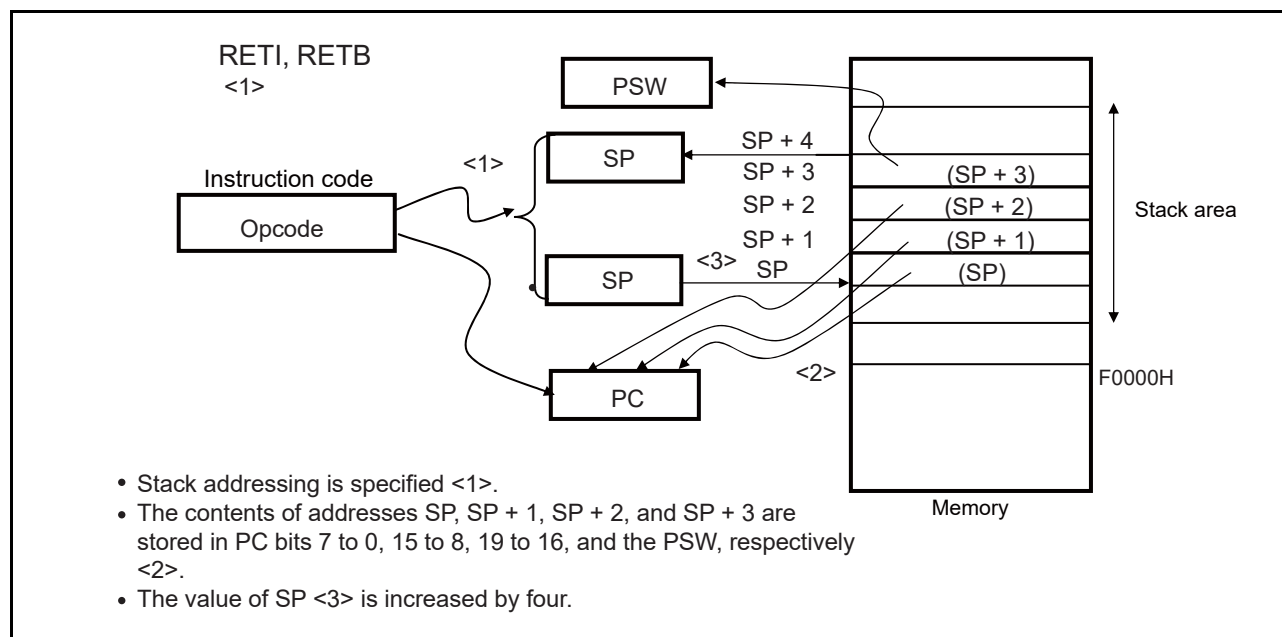


Figure 4 - 28 Example of RETI and RETB



Section 5 Instruction Set

This section lists the instructions included in the instruction set of the RL78 microcontroller. In general, the instructions are common to RL78 microcontrollers. However, the instruction set of the RL78-S1 core does not include the following CPU control instruction.

- SEL RBn (Register Bank Selection)

The following multiply, divide, and multiply-and-accumulate instructions are extended instructions and only included in the instruction set of the RL78-S3 core. For details, refer to user's manual of each product.

- MULHU (16-bit multiplication unsigned)
- MULH (16-bit multiplication signed)
- DIVHU (16-bit division unsigned)
- DIVWU (32-bit division unsigned)
- MACHU (16-bit multiplication and accumulation unsigned (16 bits × 16 bits) + 32 bits)
- MACH (16-bit multiplication and accumulation signed (16 bits × 16 bits) + 32 bits)

The number of clock cycles for the following instructions differs between the RL78-S1 core and the other CPU cores. For details, see **5.5 List of Operations**.

- 16-bit data transfer instructions (MOVW, XCHW, ONEW, CLRW)
- 16-bit operation instructions (ADDW, SUBW, CMPW)
- Multiplication instruction (MULU)
- 16-bit increment and decrement instructions (INCW, DECW)
- 16-bit shift instructions (SHRW, SHLW, SARW)
- 16-bit shift rotate instruction (ROLWC)
- Call and return instructions (CALL, CALLT, BRK, RET, RETI, RETB)
- Stack manipulation instructions (PUSH, POP, MOVW, ADDW, SUBW)

5.1 Operand Identifiers and Description Methods

Operands are described in the "Operand" column of each instruction in accordance with the description method of the instruction operand identifier (refer to the assembler specifications for details). When there are two or more description methods, select one of them. Alphabetic letters in capitals and the symbols, #, !, !!, \$, \$!, [], and ES: are keywords and are described as they are. Each symbol has the following meaning.

- #: Immediate data specification
- !: 16-bit absolute address specification
- !!: 20-bit absolute address specification
- \$: 8-bit relative address specification
- \$!: 16-bit relative address specification
- []: Indirect address specification
- ES:: Extension address specification

In the case of immediate data, describe an appropriate numeric value or a label. When using a label, be sure to describe the #, !, !!, \$, \$!, [], and ES: symbols. For operand register identifiers, r and rp, either function names (X, A, C, etc.) or absolute names (names in parentheses in **Table 5 - 1**, R0, R1, R2, etc.) can be used for description.

Table 5 - 1 Operand Identifiers and Description Methods

Identifier	Description Method
r rp sfr sfrp	X (R0), A (R1), C (R2), B (R3), E (R4), D (R5), L (R6), H (R7) AX (RP0), BC (RP1), DE (RP2), HL (RP3) Special-function register symbol (SFR symbol) FFF00H to FFFFFH Special-function register symbols (16-bit manipulatable SFR symbol. Even addresses only ^{Note 1}) FFF00H to FFFFFH
saddr saddrp	FFE20H to FFF1FH Immediate data or labels FFE20H to FF1FH Immediate data or labels (even addresses only ^{Note 1})
addr20 addr16 addr5	00000H to FFFFFH Immediate data or labels 0000H to FFFFH Immediate data or labels (only even addresses for 16-bit data transfer instructions ^{Note 1}) 0080H to 00BFH Immediate data or labels (even addresses only)
word byte bit	16-bit immediate data or label 8-bit immediate data or label 3-bit immediate data or label
RBn ^{Note 2}	RB0 to RB3

Note 1. Bit 0 = 0 when an odd address is specified.

Note 2. The RL78-S1 core does not include these registers.

Remark The special function registers can be described to operand sfr as symbols. The extended special function registers can be described to operand !addr16 as symbols.

5.2 Description in “Operation” Column

The operation when the instruction is executed is shown in the “Operation” column using the following symbols.

Table 5 - 2 Symbols in “Operation” Column

Symbol	Function
A	A register; 8-bit accumulator
X	X register
B	B register
C	C register
D	D register
E	E register
H	H register
L	L register
ES	ES register
CS	CS register
AX	AX register pair; 16-bit accumulator
BC	BC register pair
DE	DE register pair
HL	HL register pair
PC	Program counter
SP	Stack pointer
PSW	Program status word
CY	Carry flag
AC	Auxiliary carry flag
Z	Zero flag
RBS ^{Note}	Register bank select flag
IE	Interrupt request enable flag
()	Memory contents indicated by address or register contents in parentheses
XH, XL Xs, XH, XL	16-bit registers: XH = higher 8 bits, XL = lower 8 bits 20-bit registers: Xs = (bits 19 to 16), XH = (bits 15 to 8), XL = (bits 7 to 0)
^	Logical product (AND)
∨	Logical sum (OR)
⊕	Exclusive logical sum (exclusive OR)
—	Inverted data
addr5	16-bit immediate data (even addresses only in 0080H to 00BFH)
addr16	16-bit immediate data
addr20	20-bit immediate data
jdisp8	Signed 8-bit data (displacement value)
jdisp16	Signed 16-bit data (displacement value)

Note The RL78-S1 core does not include these flags.

5.3 Description in “Flag” Column

The change of the flag value when the instruction is executed is shown in the “Flag” column using the following symbols.

Table 5 - 3 Symbols in “Flag” Column

Symbol	Change of Flag Value
(Blank)	Unchanged
0	Cleared to 0
1	Set to 1
x	Set/cleared according to the result
R	Previously saved value is restored

5.4 PREFIX Instruction

Instructions with “ES:” have a PREFIX operation code as a prefix to extend the accessible data area to the 1 MB space (00000H to FFFFFH), by adding the ES register value to the 64 KB space from F0000H to FFFFFH. When a PREFIX operation code is attached as a prefix to the target instruction, only one instruction immediately after the PREFIX operation code is executed as the addresses with the ES register value added. An interrupt and DMA transfer are not acknowledged between a PREFIX instruction code and the instruction immediately after.

Table 5 - 4 Use Example of PREFIX Operation Code

Instruction	Opcode				
	1	2	3	4	5
MOV !addr16, #byte	CFH	!addr16		#byte	—
MOV ES:!addr16, #byte	11H	CFH	!addr16		#byte
MOV A, [HL]	8BH	—	—	—	—
MOV A, ES:[HL]	11H	8BH	—	—	—

Caution Set the ES register value with MOV ES, A, etc., before executing the PREFIX instruction.

5.5 List of Operations

5.5.1 Operation list of RL78-S1 core

Table 5 - 5 Operation List of RL78-S1 Core (1/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	r, #byte	2	1	—	$r \leftarrow \text{byte}$			
		PSW, #byte	3	3	—	$\text{PSW} \leftarrow \text{byte}$	x	x	x
		CS, #byte	3	1	—	$\text{CS} \leftarrow \text{byte}$			
		ES, #byte	2	1	—	$\text{ES} \leftarrow \text{byte}$			
		!addr16, #byte	4	1	—	$(\text{addr16}) \leftarrow \text{byte}$			
		ES:!addr16, #byte	5	2	—	$(\text{ES}, \text{addr16}) \leftarrow \text{byte}$			
		saddr, #byte	3	1	—	$(\text{saddr}) \leftarrow \text{byte}$			
		sfr, #byte	3	1	—	$\text{sfr} \leftarrow \text{byte}$			
		[DE+byte], #byte	3	1	—	$(\text{DE} + \text{byte}) \leftarrow \text{byte}$			
		ES:[DE+byte], #byte	4	2	—	$((\text{ES}, \text{DE}) + \text{byte}) \leftarrow \text{byte}$			
		[HL+byte], #byte	3	1	—	$(\text{HL} + \text{byte}) \leftarrow \text{byte}$			
		ES:[HL+byte], #byte	4	2	—	$((\text{ES}, \text{HL}) + \text{byte}) \leftarrow \text{byte}$			
		[SP+byte], #byte	3	1	—	$(\text{SP} + \text{byte}) \leftarrow \text{byte}$			
		word[B], #byte	4	1	—	$(\text{B} + \text{word}) \leftarrow \text{byte}$			
		ES:word[B], #byte	5	2	—	$((\text{ES}, \text{B}) + \text{word}) \leftarrow \text{byte}$			
		word[C], #byte	4	1	—	$(\text{C} + \text{word}) \leftarrow \text{byte}$			
		ES:word[C], #byte	5	2	—	$((\text{ES}, \text{C}) + \text{word}) \leftarrow \text{byte}$			
		word[BC], #byte	4	1	—	$(\text{BC} + \text{word}) \leftarrow \text{byte}$			
		ES:word[BC], #byte	5	2	—	$((\text{ES}, \text{BC}) + \text{word}) \leftarrow \text{byte}$			
		A, r ^{Note 3}	1	1	—	$\text{A} \leftarrow r$			
		r ^{Note 3} , A	1	1	—	$r \leftarrow \text{A}$			
		A, PSW	2	1	—	$\text{A} \leftarrow \text{PSW}$			
		PSW, A	2	3	—	$\text{PSW} \leftarrow \text{A}$	x	x	x
		A, CS	2	1	—	$\text{A} \leftarrow \text{CS}$			
		CS, A	2	1	—	$\text{CS} \leftarrow \text{A}$			
		A, ES	2	1	—	$\text{A} \leftarrow \text{ES}$			
		ES, A	2	1	—	$\text{ES} \leftarrow \text{A}$			
		A, !addr16	3	1	4	$\text{A} \leftarrow (\text{addr16})$			
		A, ES:!addr16	4	2	5	$\text{A} \leftarrow (\text{ES}, \text{addr16})$			
		!addr16, A	3	1	—	$(\text{addr16}) \leftarrow \text{A}$			
		ES:!addr16, A	4	2	—	$(\text{ES}, \text{addr16}) \leftarrow \text{A}$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (2/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	A, saddr	2	1	—	$A \leftarrow (\text{saddr})$			
		saddr, A	2	1	—	$(\text{saddr}) \leftarrow A$			
		A, sfr	2	1	—	$A \leftarrow \text{sfr}$			
		sfr, A	2	1	—	$\text{sfr} \leftarrow A$			
		A, [DE]	1	1	4	$A \leftarrow (\text{DE})$			
		[DE], A	1	1	—	$(\text{DE}) \leftarrow A$			
		A, ES:[DE]	2	2	5	$A \leftarrow (\text{ES}, \text{DE})$			
		ES:[DE], A	2	2	—	$(\text{ES}, \text{DE}) \leftarrow A$			
		A, [HL]	1	1	4	$A \leftarrow (\text{HL})$			
		[HL], A	1	1	—	$(\text{HL}) \leftarrow A$			
		A, ES:[HL]	2	2	5	$A \leftarrow (\text{ES}, \text{HL})$			
		ES:[HL], A	2	2	—	$(\text{ES}, \text{HL}) \leftarrow A$			
		A, [DE+byte]	2	1	4	$A \leftarrow (\text{DE} + \text{byte})$			
		[DE+byte], A	2	1	—	$(\text{DE} + \text{byte}) \leftarrow A$			
		A, ES:[DE+byte]	3	2	5	$A \leftarrow ((\text{ES}, \text{DE}) + \text{byte})$			
		ES:[DE+byte], A	3	2	—	$((\text{ES}, \text{DE}) + \text{byte}) \leftarrow A$			
		A, [HL+byte]	2	1	4	$A \leftarrow (\text{HL} + \text{byte})$			
		[HL+byte], A	2	1	—	$(\text{HL} + \text{byte}) \leftarrow A$			
		A, ES:[HL+byte]	3	2	5	$A \leftarrow ((\text{ES}, \text{HL}) + \text{byte})$			
		ES:[HL+byte], A	3	2	—	$((\text{ES}, \text{HL}) + \text{byte}) \leftarrow A$			
		A, [SP+byte]	2	1	—	$A \leftarrow (\text{SP} + \text{byte})$			
		[SP+byte], A	2	1	—	$(\text{SP} + \text{byte}) \leftarrow A$			
		A, word[B]	3	1	4	$A \leftarrow (\text{B} + \text{word})$			
		word[B], A	3	1	—	$(\text{B} + \text{word}) \leftarrow A$			
		A, ES:word[B]	4	2	5	$A \leftarrow ((\text{ES}, \text{B}) + \text{word})$			
		ES:word[B], A	4	2	—	$((\text{ES}, \text{B}) + \text{word}) \leftarrow A$			
		A, word[C]	3	1	4	$A \leftarrow (\text{C} + \text{word})$			
		word[C], A	3	1	—	$(\text{C} + \text{word}) \leftarrow A$			
		A, ES:word[C]	4	2	5	$A \leftarrow ((\text{ES}, \text{C}) + \text{word})$			
		ES:word[C], A	4	2	—	$((\text{ES}, \text{C}) + \text{word}) \leftarrow A$			
		A, word[BC]	3	1	4	$A \leftarrow (\text{BC} + \text{word})$			
		word[BC], A	3	1	—	$(\text{BC} + \text{word}) \leftarrow A$			
		A, ES:word[BC]	4	2	5	$A \leftarrow ((\text{ES}, \text{BC}) + \text{word})$			
		ES:word[BC], A	4	2	—	$((\text{ES}, \text{BC}) + \text{word}) \leftarrow A$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (3/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	A, [HL+B]	2	1	4	$A \leftarrow (HL + B)$			
		[HL+B], A	2	1	—	$(HL + B) \leftarrow A$			
		A, ES:[HL+B]	3	2	5	$A \leftarrow ((ES, HL) + B)$			
		ES:[HL+B], A	3	2	—	$((ES, HL) + B) \leftarrow A$			
		A, [HL+C]	2	1	4	$A \leftarrow (HL + C)$			
		[HL+C], A	2	1	—	$(HL + C) \leftarrow A$			
		A, ES:[HL+C]	3	2	5	$A \leftarrow ((ES, HL) + C)$			
		ES:[HL+C], A	3	2	—	$((ES, HL) + C) \leftarrow A$			
		X, !addr16	3	1	4	$X \leftarrow (addr16)$			
		X, ES:!addr16	4	2	5	$X \leftarrow (ES, addr16)$			
		X, saddr	2	1	—	$X \leftarrow (saddr)$			
		B, !addr16	3	1	4	$B \leftarrow (addr16)$			
		B, ES:!addr16	4	2	5	$B \leftarrow (ES, addr16)$			
		B, saddr	2	1	—	$B \leftarrow (saddr)$			
		C, !addr16	3	1	4	$C \leftarrow (addr16)$			
		C, ES:!addr16	4	2	5	$C \leftarrow (ES, addr16)$			
		C, saddr	2	1	—	$C \leftarrow (saddr)$			
		ES, saddr	3	1	—	$ES \leftarrow (saddr)$			
	XCH	A, r ^{Note 3}	1 (r = X) 2 (other than r = X)	1	—	$A \longleftrightarrow r$			
		A, !addr16	4	2	—	$A \longleftrightarrow (addr16)$			
		A, ES:!addr16	5	3	—	$A \longleftrightarrow (ES, addr16)$			
		A, saddr	3	2	—	$A \longleftrightarrow (saddr)$			
		A, sfr	3	2	—	$A \longleftrightarrow sfr$			
		A, [DE]	2	2	—	$A \longleftrightarrow (DE)$			
		A, ES:[DE]	3	3	—	$A \longleftrightarrow (ES, DE)$			
		A, [HL]	2	2	—	$A \longleftrightarrow (HL)$			
		A, ES:[HL]	3	3	—	$A \longleftrightarrow (ES, HL)$			
		A, [DE+byte]	3	2	—	$A \longleftrightarrow (DE + \text{byte})$			
		A, ES:[DE+byte]	4	3	—	$A \longleftrightarrow ((ES, DE) + \text{byte})$			
		A, [HL+byte]	3	2	—	$A \longleftrightarrow (HL + \text{byte})$			
		A, ES:[HL+byte]	4	3	—	$A \longleftrightarrow ((ES, HL) + \text{byte})$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (4/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	XCH	A, [HL+B]	2	2	—	$A \leftrightarrow (HL + B)$			
		A, ES:[HL+B]	3	3	—	$A \leftrightarrow ((ES, HL) + B)$			
		A, [HL+C]	2	2	—	$A \leftrightarrow (HL + C)$			
		A, ES:[HL+C]	3	3	—	$A \leftrightarrow ((ES, HL) + C)$			
	ONEB	A	1	1	—	$A \leftarrow 01H$			
		X	1	1	—	$X \leftarrow 01H$			
		B	1	1	—	$B \leftarrow 01H$			
		C	1	1	—	$C \leftarrow 01H$			
		!addr16	3	1	—	$(addr16) \leftarrow 01H$			
		ES:!addr16	4	2	—	$(ES, addr16) \leftarrow 01H$			
		saddr	2	1	—	$(saddr) \leftarrow 01H$			
	CLRB	A	1	1	—	$A \leftarrow 00H$			
		X	1	1	—	$X \leftarrow 00H$			
		B	1	1	—	$B \leftarrow 00H$			
		C	1	1	—	$C \leftarrow 00H$			
		!addr16	3	1	—	$(addr16) \leftarrow 00H$			
		ES:!addr16	4	2	—	$(ES, addr16) \leftarrow 00H$			
		saddr	2	1	—	$(saddr) \leftarrow 00H$			
	MOVS	[HL+byte], X	3	1	—	$(HL + byte) \leftarrow X$	x		x
		ES:[HL+byte], X	4	2	—	$(ES, HL + byte) \leftarrow X$	x		x
16-bit data transfer	MOVW	rp, #word	3	2	—	$rp \leftarrow word$			
		saddrp, #word	4	2	—	$(saddrp) \leftarrow word$			
		sfrp, #word	4	2	—	$sfrp \leftarrow word$			
		AX, rp ^{Note 3}	1	2	—	$AX \leftarrow rp$			
		rp ^{Note 3} , AX	1	2	—	$rp \leftarrow AX$			
		AX, !addr16	3	2	5	$AX \leftarrow (addr16)$			
		!addr16, AX	3	2	—	$(addr16) \leftarrow AX$			
		AX, ES:!addr16	4	3	6	$AX \leftarrow (ES, addr16)$			
		ES:!addr16, AX	4	3	—	$(ES, addr16) \leftarrow AX$			
		AX, saddrp	2	2	—	$AX \leftarrow (saddrp)$			
		saddrp, AX	2	2	—	$(saddrp) \leftarrow AX$			
		AX, sfrp	2	2	—	$AX \leftarrow sfrp$			
		sfrp, AX	2	2	—	$sfrp \leftarrow AX$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register pair other than AX.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (5/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit data transfer	MOVW	AX, [DE]	1	2	5	$AX \leftarrow (DE)$			
		[DE], AX	1	2	—	$(DE) \leftarrow AX$			
		AX, ES:[DE]	2	3	6	$AX \leftarrow (ES, DE)$			
		ES:[DE], AX	2	3	—	$(ES, DE) \leftarrow AX$			
		AX, [HL]	1	2	5	$AX \leftarrow (HL)$			
		[HL], AX	1	2	—	$(HL) \leftarrow AX$			
		AX, ES:[HL]	2	3	6	$AX \leftarrow (ES, HL)$			
		ES:[HL], AX	2	3	—	$(ES, HL) \leftarrow AX$			
		AX, [DE+byte]	2	2	5	$AX \leftarrow (DE + \text{byte})$			
		[DE+byte], AX	2	2	—	$(DE + \text{byte}) \leftarrow AX$			
		AX, ES:[DE+byte]	3	3	6	$AX \leftarrow ((ES, DE) + \text{byte})$			
		ES:[DE+byte], AX	3	3	—	$((ES, DE) + \text{byte}) \leftarrow AX$			
		AX, [HL+byte]	2	2	5	$AX \leftarrow (HL + \text{byte})$			
		[HL+byte], AX	2	2	—	$(HL + \text{byte}) \leftarrow AX$			
		AX, ES:[HL+byte]	3	3	6	$AX \leftarrow ((ES, HL) + \text{byte})$			
		ES:[HL+byte], AX	3	3	—	$((ES, HL) + \text{byte}) \leftarrow AX$			
		AX, [SP+byte]	2	2	—	$AX \leftarrow (SP + \text{byte})$			
		[SP+byte], AX	2	2	—	$(SP + \text{byte}) \leftarrow AX$			
		AX, word[B]	3	2	5	$AX \leftarrow (B + \text{word})$			
		word[B], AX	3	2	—	$(B + \text{word}) \leftarrow AX$			
		AX, ES:word[B]	4	3	6	$AX \leftarrow ((ES, B) + \text{word})$			
		ES:word[B], AX	4	3	—	$((ES, B) + \text{word}) \leftarrow AX$			
		AX, word[C]	3	2	5	$AX \leftarrow (C + \text{word})$			
		word[C], AX	3	2	—	$(C + \text{word}) \leftarrow AX$			
		AX, ES:word[C]	4	3	6	$AX \leftarrow ((ES, C) + \text{word})$			
		ES:word[C], AX	4	3	—	$((ES, C) + \text{word}) \leftarrow AX$			
		AX, word[BC]	3	2	5	$AX \leftarrow (BC + \text{word})$			
		word[BC], AX	3	2	—	$(BC + \text{word}) \leftarrow AX$			
		AX, ES:word[BC]	4	3	6	$AX \leftarrow ((ES, BC) + \text{word})$			
		ES:word[BC], AX	4	3	—	$((ES, BC) + \text{word}) \leftarrow AX$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (6/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit data transfer	MOVW	BC, !addr16	3	2	5	$BC \leftarrow (addr16)$			
		BC, ES:!addr16	4	3	6	$BC \leftarrow (ES, addr16)$			
		DE, !addr16	3	2	5	$DE \leftarrow (addr16)$			
		DE, ES:!addr16	4	3	6	$DE \leftarrow (ES, addr16)$			
		HL, !addr16	3	2	5	$HL \leftarrow (addr16)$			
		HL, ES:!addr16	4	3	6	$HL \leftarrow (ES, addr16)$			
		BC, saddrp	2	2	—	$BC \leftarrow (saddrp)$			
		DE, saddrp	2	2	—	$DE \leftarrow (saddrp)$			
		HL, saddrp	2	2	—	$HL \leftarrow (saddrp)$			
	XCHW	AX, rp ^{Note 3}	1	2	—	$AX \leftrightarrow rp$			
	ONEW	AX	1	2	—	$AX \leftarrow 0001H$			
		BC	1	2	—	$BC \leftarrow 0001H$			
	CLRW	AX	1	2	—	$AX \leftarrow 0000H$			
		BC	1	2	—	$BC \leftarrow 0000H$			
8-bit operation	ADD	A, #byte	2	1	—	$A, CY \leftarrow A + \text{byte}$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) + \text{byte}$	x	x	x
		A, r ^{Note 4}	2	1	—	$A, CY \leftarrow A + r$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r + A$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A + (addr16)$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A + (ES, addr16)$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A + (saddr)$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A + (HL)$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A + (ES, HL)$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A + (HL + \text{byte})$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + \text{byte})$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A + (HL + B)$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + B)$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A + (HL + C)$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + C)$	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register pair other than AX.

Note 4. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (7/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	ADDC	A, #byte	2	1	—	$A, CY \leftarrow A + \text{byte} + CY$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) + \text{byte} + CY$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A + r + CY$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r + A + CY$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A + (\text{addr16}) + CY$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A + (ES, \text{addr16}) + CY$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A + (saddr) + CY$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A + (HL) + CY$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A + (ES, HL) + CY$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A + (HL + \text{byte}) + CY$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + \text{byte}) + CY$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A + (HL + B) + CY$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + B) + CY$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A + (HL + C) + CY$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + C) + CY$	x	x	x
	SUB	A, #byte	2	1	—	$A, CY \leftarrow A - \text{byte}$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) - \text{byte}$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A - r$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r - A$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A - (\text{addr16})$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A - (ES, \text{addr16})$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A - (saddr)$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A - (HL)$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A - (ES, HL)$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A - (HL + \text{byte})$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + \text{byte})$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A - (HL + B)$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + B)$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A - (HL + C)$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + C)$	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (8/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	SUBC	A, #byte	2	1	—	$A, CY \leftarrow A - \text{byte} - CY$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) - \text{byte} - CY$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A - r - CY$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r - A - CY$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A - (\text{addr16}) - CY$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A - (\text{ES}, \text{addr16}) - CY$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A - (saddr) - CY$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A - (\text{HL}) - CY$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A - (\text{ES}, \text{HL}) - CY$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A - (\text{HL} + \text{byte}) - CY$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A - ((\text{ES}, \text{HL}) + \text{byte}) - CY$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A - (\text{HL} + B) - CY$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A - ((\text{ES}, \text{HL}) + B) - CY$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A - (\text{HL} + C) - CY$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A - ((\text{ES}, \text{HL}) + C) - CY$	x	x	x
	AND	A, #byte	2	1	—	$A \leftarrow A \wedge \text{byte}$	x		
		saddr, #byte	3	2	—	$(saddr) \leftarrow (saddr) \wedge \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \wedge r$	x		
		r, A	2	1	—	$r \leftarrow r \wedge A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \wedge (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \wedge (\text{ES}, \text{addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \wedge (saddr)$	x		
		A, [HL]	1	1	4	$A \leftarrow A \wedge (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \wedge (\text{ES}, \text{HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \wedge (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \wedge ((\text{ES}, \text{HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \wedge (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \wedge ((\text{ES}, \text{HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \wedge (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \wedge ((\text{ES}, \text{HL}) + C)$	x		

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (9/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	OR	A, #byte	2	1	—	$A \leftarrow A \vee \text{byte}$	x		
		saddr, #byte	3	2	—	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \vee r$	x		
		r, A	2	1	—	$r \leftarrow r \vee A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \vee (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \vee (\text{ES:addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \vee (\text{saddr})$	x		
		A, [HL]	1	1	4	$A \leftarrow A \vee (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \vee (\text{ES:HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \vee (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \vee (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \vee (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + C)$	x		
	XOR	A, #byte	2	1	—	$A \leftarrow A \vee \text{byte}$	x		
		saddr, #byte	3	2	—	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \vee r$	x		
		r, A	2	1	—	$r \leftarrow r \vee A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \vee (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \vee (\text{ES:addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \vee (\text{saddr})$	x		
		A, [HL]	1	1	4	$A \leftarrow A \vee (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \vee (\text{ES:HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \vee (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \vee (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \vee (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + C)$	x		

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (10/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	CMP	A, #byte	2	1	—	A - byte	x	x	x
		!addr16, #byte	4	1	4	(addr16) - byte	x	x	x
		ES:!addr16, #byte	5	2	5	(ES:addr16) - byte	x	x	x
		saddr, #byte	3	1	—	(saddr) - byte	x	x	x
		A, r ^{Note 3}	2	1	—	A - r	x	x	x
		r, A	2	1	—	r - A	x	x	x
		A, !addr16	3	1	4	A - (addr16)	x	x	x
		A, ES:!addr16	4	2	5	A - (ES:addr16)	x	x	x
		A, saddr	2	1	—	A - (saddr)	x	x	x
		A, [HL]	1	1	4	A - (HL)	x	x	x
		A, ES:[HL]	2	2	5	A - (ES:HL)	x	x	x
		A, [HL+byte]	2	1	4	A - (HL + byte)	x	x	x
		A, ES:[HL+byte]	3	2	5	A - ((ES:HL) + byte)	x	x	x
		A, [HL+B]	2	1	4	A - (HL + B)	x	x	x
		A, ES:[HL+B]	3	2	5	A - ((ES:HL) + B)	x	x	x
		A, [HL+C]	2	1	4	A - (HL + C)	x	x	x
		A, ES:[HL+C]	3	2	5	A - ((ES:HL) + C)	x	x	x
	CMP0	A	1	1	—	A - 00H	x	0	0
		X	1	1	—	X - 00H	x	0	0
		B	1	1	—	B - 00H	x	0	0
		C	1	1	—	C - 00H	x	0	0
		!addr16	3	1	4	(addr16) - 00H	x	0	0
		ES:!addr16	4	2	5	(ES:addr16) - 00H	x	0	0
		saddr	2	1	—	(saddr) - 00H	x	0	0
	CMPS	X, [HL+byte]	3	1	4	X - (HL + byte)	x	x	x
		X, ES:[HL+byte]	4	2	5	X - ((ES:HL) + byte)	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (11/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit operation	ADDW	AX, #word	3	2	—	$AX, CY \leftarrow AX + \text{word}$	x	x	x
		AX, AX	1	2	—	$AX, CY \leftarrow AX + AX$	x	x	x
		AX, BC	1	2	—	$AX, CY \leftarrow AX + BC$	x	x	x
		AX, DE	1	2	—	$AX, CY \leftarrow AX + DE$	x	x	x
		AX, HL	1	2	—	$AX, CY \leftarrow AX + HL$	x	x	x
		AX, !addr16	3	2	5	$AX, CY \leftarrow AX + (\text{addr16})$	x	x	x
		AX, ES:!addr16	4	3	6	$AX, CY \leftarrow AX + (\text{ES:addr16})$	x	x	x
		AX, saddrp	2	2	—	$AX, CY \leftarrow AX + (\text{saddrp})$	x	x	x
		AX, [HL+byte]	3	2	5	$AX, CY \leftarrow AX + (\text{HL} + \text{byte})$	x	x	x
		AX, ES:[HL+byte]	4	3	6	$AX, CY \leftarrow AX + ((\text{ES:HL}) + \text{byte})$	x	x	x
	SUBW	AX, #word	3	2	—	$AX, CY \leftarrow AX - \text{word}$	x	x	x
		AX, BC	1	2	—	$AX, CY \leftarrow AX - BC$	x	x	x
		AX, DE	1	2	—	$AX, CY \leftarrow AX - DE$	x	x	x
		AX, HL	1	2	—	$AX, CY \leftarrow AX - HL$	x	x	x
		AX, !addr16	3	2	5	$AX, CY \leftarrow AX - (\text{addr16})$	x	x	x
		AX, ES:!addr16	4	3	6	$AX, CY \leftarrow AX - (\text{ES:addr16})$	x	x	x
		AX, saddrp	2	2	—	$AX, CY \leftarrow AX - (\text{saddrp})$	x	x	x
		AX, [HL+byte]	3	2	5	$AX, CY \leftarrow AX - (\text{HL} + \text{byte})$	x	x	x
		AX, ES:[HL+byte]	4	3	6	$AX, CY \leftarrow AX - ((\text{ES:HL}) + \text{byte})$	x	x	x
	CMPW	AX, #word	3	2	—	$AX - \text{word}$	x	x	x
		AX, BC	1	2	—	$AX - BC$	x	x	x
		AX, DE	1	2	—	$AX - DE$	x	x	x
		AX, HL	1	2	—	$AX - HL$	x	x	x
		AX, !addr16	3	2	5	$AX - (\text{addr16})$	x	x	x
		AX, ES:!addr16	4	3	6	$AX - (\text{ES:addr16})$	x	x	x
		AX, saddrp	2	2	—	$AX - (\text{saddrp})$	x	x	x
		AX, [HL+byte]	3	2	5	$AX - (\text{HL} + \text{byte})$	x	x	x
		AX, ES:[HL+byte]	4	3	6	$AX - ((\text{ES:HL}) + \text{byte})$	x	x	x
Multiply	MULU	X	1	2	—	$AX \leftarrow A \times X$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (12/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
<R>	Increment and decrement	r	1	1	—	$r \leftarrow r + 1$	x	x	
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) + 1$	x	x	
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) + 1$	x	x	
		saddr	2	2	—	$(saddr) \leftarrow (saddr) + 1$	x	x	
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) + 1$	x	x	
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) + 1$	x	x	
		DEC	r	1	1	$r \leftarrow r - 1$	x	x	
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) - 1$	x	x	
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) - 1$	x	x	
		saddr	2	2	—	$(saddr) \leftarrow (saddr) - 1$	x	x	
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) - 1$	x	x	
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) - 1$	x	x	
	INCW	rp	1	2	—	$rp \leftarrow rp + 1$			
		!addr16	3	4	—	$(addr16) \leftarrow (addr16) + 1$			
		ES:!addr16	4	5	—	$(ES, addr16) \leftarrow (ES, addr16) + 1$			
		saddrp	2	4	—	$(saddrp) \leftarrow (saddrp) + 1$			
		[HL+byte]	3	4	—	$(HL + byte) \leftarrow (HL + byte) + 1$			
		ES:[HL+byte]	4	5	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) + 1$			
	DECW	rp	1	2	—	$rp \leftarrow rp - 1$			
		!addr16	3	4	—	$(addr16) \leftarrow (addr16) - 1$			
		ES:!addr16	4	5	—	$(ES, addr16) \leftarrow (ES, addr16) - 1$			
		saddrp	2	4	—	$(saddrp) \leftarrow (saddrp) - 1$			
		[HL+byte]	3	4	—	$(HL + byte) \leftarrow (HL + byte) - 1$			
		ES:[HL+byte]	4	5	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) - 1$			
	Shift	SHR	A, cnt	2	1	$(CY \leftarrow A_0, A_{m-1} \leftarrow A_m, A_7 \leftarrow 0) \times cnt$			x
		SHRW	AX, cnt	2	2	$(CY \leftarrow AX_0, AX_{m-1} \leftarrow AX_m, AX_{15} \leftarrow 0) \times cnt$			x
		SHL	A, cnt	2	1	$(CY \leftarrow A_7, A_m \leftarrow A_{m-1}, A_0 \leftarrow 0) \times cnt$			x
			B, cnt	2	1	$(CY \leftarrow B_7, B_m \leftarrow B_{m-1}, B_0 \leftarrow 0) \times cnt$			x
			C, cnt	2	1	$(CY \leftarrow C_7, C_m \leftarrow C_{m-1}, C_0 \leftarrow 0) \times cnt$			x
		SHLW	AX, cnt	2	2	$(CY \leftarrow AX_{15}, AX_m \leftarrow AX_{m-1}, AX_0 \leftarrow 0) \times cnt$			x
			BC, cnt	2	2	$(CY \leftarrow BC_{15}, BC_m \leftarrow BC_{m-1}, BC_0 \leftarrow 0) \times cnt$			x
		SAR	A, cnt	2	1	$(CY \leftarrow A_0, A_{m-1} \leftarrow A_m, A_7 \leftarrow A_7) \times cnt$			x
		SARW	AX, cnt	2	2	$(CY \leftarrow AX_0, AX_{m-1} \leftarrow AX_m, AX_{15} \leftarrow AX_{15}) \times cnt$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark 1. The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Remark 2. cnt indicates the bit shift count.

Table 5 - 5 Operation List of RL78-S1 Core (13/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Rotate	ROR	A, 1	2	1	—	$(CY, A7 \leftarrow A0, A_{m-1} \leftarrow A_m) \times 1$			x
	ROL	A, 1	2	1	—	$(CY, A0 \leftarrow A7, A_{m+1} \leftarrow A_m) \times 1$			x
	RORC	A, 1	2	1	—	$(CY \leftarrow A0, A7 \leftarrow CY, A_{m-1} \leftarrow A_m) \times 1$			x
	ROLC	A, 1	2	1	—	$(CY \leftarrow A7, A0 \leftarrow CY, A_{m+1} \leftarrow A_m) \times 1$			x
	ROLWC	AX, 1	2	2	—	$(CY \leftarrow AX_{15}, AX0 \leftarrow CY, AX_{m+1} \leftarrow AX_m) \times 1$			x
		BC, 1	2	2	—	$(CY \leftarrow BC_{15}, BC0 \leftarrow CY, BC_{m+1} \leftarrow BC_m) \times 1$			x
Bit manipulation	MOV1	CY, A.bit	2	1	—	$CY \leftarrow A.bit$			x
		A.bit, CY	2	1	—	$A.bit \leftarrow CY$			
		CY, PSW.bit	3	1	—	$CY \leftarrow PSW.bit$			x
		PSW.bit, CY	3	4	—	$PSW.bit \leftarrow CY$	x	x	
		CY, saddr.bit	3	1	—	$CY \leftarrow (saddr).bit$			x
		saddr.bit, CY	3	2	—	$(saddr).bit \leftarrow CY$			
		CY, sfr.bit	3	1	—	$CY \leftarrow sfr.bit$			x
		sfr.bit, CY	3	2	—	$sfr.bit \leftarrow CY$			
		CY, [HL].bit	2	1	4	$CY \leftarrow (HL).bit$			x
		[HL].bit, CY	2	2	—	$(HL).bit \leftarrow CY$			
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow (ES, HL).bit$			x
		ES:[HL].bit, CY	3	3	—	$(ES, HL).bit \leftarrow CY$			
	AND1	CY, A.bit	2	1	—	$CY \leftarrow CY \wedge A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \wedge PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \wedge (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \wedge sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \wedge (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \wedge (ES, HL).bit$			x
	OR1	CY, A.bit	2	1	—	$CY \leftarrow CY \vee A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \vee PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \vee (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \vee sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \vee (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \vee (ES, HL).bit$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (14/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Bit manipulation	XOR1	CY, A.bit	2	1	—	$CY \leftarrow CY \vee A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \vee PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \vee (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \vee sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \vee (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \vee (ES, HL).bit$			x
	SET1	A.bit	2	1	—	$A.bit \leftarrow 1$			
		PSW.bit	3	4	—	$PSW.bit \leftarrow 1$	x	x	x
		!addr16.bit	4	2	—	$(addr16).bit \leftarrow 1$			
		ES:!addr16.bit	5	3	—	$(ES, addr16).bit \leftarrow 1$			
		saddr.bit	3	2	—	$(saddr).bit \leftarrow 1$			
		sfr.bit	3	2	—	$sfr.bit \leftarrow 1$			
		[HL].bit	2	2	—	$(HL).bit \leftarrow 1$			
		ES:[HL].bit	3	3	—	$(ES, HL).bit \leftarrow 1$			
	CLR1	A.bit	2	1	—	$A.bit \leftarrow 0$			
		PSW.bit	3	4	—	$PSW.bit \leftarrow 0$	x	x	x
		!addr16.bit	4	2	—	$(addr16).bit \leftarrow 0$			
		ES:!addr16.bit	5	3	—	$(ES, addr16).bit \leftarrow 0$			
		saddr.bit	3	2	—	$(saddr).bit \leftarrow 0$			
		sfr.bit	3	2	—	$sfr.bit \leftarrow 0$			
		[HL].bit	2	2	—	$(HL).bit \leftarrow 0$			
		ES:[HL].bit	3	3	—	$(ES, HL).bit \leftarrow 0$			
	SET1	CY	2	1	—	$CY \leftarrow 1$			1
	CLR1	CY	2	1	—	$CY \leftarrow 0$			0
	NOT1	CY	2	1	—	$CY \leftarrow \overline{CY}$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (15/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Call and return	CALL	rp	2	4	—	$(SP - 2) \leftarrow (PC + 2)_S, (SP - 3) \leftarrow (PC + 2)_H,$ $(SP - 4) \leftarrow (PC + 2)_L, PC \leftarrow CS, rp,$ $SP \leftarrow SP - 4$			
		\$!addr20	3	4	—	$(SP - 2) \leftarrow (PC + 3)_S, (SP - 3) \leftarrow (PC + 3)_H,$ $(SP - 4) \leftarrow (PC + 3)_L, PC \leftarrow PC + 3 + jdisp16,$ $SP \leftarrow SP - 4$			
		!addr16	3	4	—	$(SP - 2) \leftarrow (PC + 3)_S, (SP - 3) \leftarrow (PC + 3)_H,$ $(SP - 4) \leftarrow (PC + 3)_L, PC \leftarrow 0000, addr16,$ $SP \leftarrow SP - 4$			
		!!addr20	4	4	—	$(SP - 2) \leftarrow (PC + 4)_S, (SP - 3) \leftarrow (PC + 4)_H,$ $(SP - 4) \leftarrow (PC + 4)_L, PC \leftarrow addr20,$ $SP \leftarrow SP - 4$			
	CALLT	[addr5]	2	6	—	$(SP - 2) \leftarrow (PC + 2)_S, (SP - 3) \leftarrow (PC + 2)_H,$ $(SP - 4) \leftarrow (PC + 2)_L, PCs \leftarrow 0000,$ $PCH \leftarrow (0000, addr5 + 1),$ $PCL \leftarrow (0000, addr5),$ $SP \leftarrow SP - 4$			
	BRK	—	2	7	—	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow (PC + 2)_S,$ $(SP - 3) \leftarrow (PC + 2)_H, (SP - 4) \leftarrow (PC + 2)_L,$ $PCs \leftarrow 0000,$ $PCH \leftarrow (0007FH), PCL \leftarrow (0007EH),$ $SP \leftarrow SP - 4, IE \leftarrow 0$			
	RET	—	1	7	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), SP \leftarrow SP + 4$			
	RETI	—	2	8	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), PSW \leftarrow (SP + 3),$ $SP \leftarrow SP + 4$	R	R	R
	RETB	—	2	8	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), PSW \leftarrow (SP + 3),$ $SP \leftarrow SP + 4$	R	R	R

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (16/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Stack manipulation	PUSH	PSW	2	2	—	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow 00H, SP \leftarrow SP - 2$			
		rp	1	2	—	$(SP - 1) \leftarrow rpH, (SP - 2) \leftarrow rpL, SP \leftarrow SP - 2$			
	POP	PSW	2	4	—	$PSW \leftarrow (SP + 1), SP \leftarrow SP + 2$	R	R	R
		rp	1	2	—	$rpL \leftarrow (SP), rpH \leftarrow (SP + 1), SP \leftarrow SP + 2$			
	MOVW	SP, #word	4	2	—	$SP \leftarrow word$			
		SP, AX	2	2	—	$SP \leftarrow AX$			
		AX, SP	2	2	—	$AX \leftarrow SP$			
		HL, SP	3	2	—	$HL \leftarrow SP$			
		BC, SP	3	2	—	$BC \leftarrow SP$			
		DE, SP	3	2	—	$DE \leftarrow SP$			
	ADDW	SP, #byte	2	2	—	$SP \leftarrow SP + byte$			
	SUBW	SP, #byte	2	2	—	$SP \leftarrow SP - byte$			
Unconditional branch	BR	AX	2	3	—	$PC \leftarrow CS, AX$			
		\$addr20	2	3	—	$PC \leftarrow PC + 2 + jdisp8$			
		\$!addr20	3	3	—	$PC \leftarrow PC + 3 + jdisp16$			
		!addr16	3	3	—	$PC \leftarrow 0000, addr16$			
		!addr20	4	3	—	$PC \leftarrow addr20$			
Conditional branch	BC	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 1$			
	BNC	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 0$			
	BZ	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 1$			
	BNZ	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 0$			
	BH	\$addr20	3	2/4 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if $(Z \vee CY) = 0$			
	BNH	\$addr20	3	2/4 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if $(Z \vee CY) = 1$			
	BT	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 1			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 1			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 1			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 1			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. These indicate the numbers of clock cycles when the condition is not and is met.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

Table 5 - 5 Operation List of RL78-S1 Core (17/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Conditional branch	BT	[HL].bit, \$addr20	3	3/5 Note 3	6/7	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 1			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	7/8	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 1			
	BF	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 0			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 0			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 0			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 0			
		[HL].bit, \$addr20	3	3/5 Note 3	6/7	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 0			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	7/8	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 0			
	BTCLR	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 1 then reset (saddr).bit			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 1 then reset sfr.bit			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 1 then reset A.bit			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 1 then reset PSW.bit	x	x	x
		[HL].bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 1 then reset (HL).bit			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 1 then reset (ES, HL).bit			
Conditional skip	SKC	—	2	1	—	Next instruction skip if CY = 1			
	SKNC	—	2	1	—	Next instruction skip if CY = 0			
	SKZ	—	2	1	—	Next instruction skip if Z = 1			
	SKNZ	—	2	1	—	Next instruction skip if Z = 0			
	SKH	—	2	1	—	Next instruction skip if $(Z \vee CY) = 0$			
	SKNH	—	2	1	—	Next instruction skip if $(Z \vee CY) = 1$			
CPU control	NOP	—	1	1	—	No Operation			
	EI	—	3	4	—	$IE \leftarrow 1$ (Enable Interrupt)			
	DI	—	3	4	—	$IE \leftarrow 0$ (Disable Interrupt)			
	HALT	—	2	3	—	Set HALT Mode			
	STOP	—	2	3	—	Set STOP Mode			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area.

Note 3. These indicate the numbers of clock cycles when the condition is not and is met.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to four times the listed number plus 6 clock cycles.

5.5.2 Operation list of RL78-S2 core

Table 5 - 6 Operation List of RL78-S2 Core (1/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	r, #byte	2	1	—	$r \leftarrow \text{byte}$			
		PSW, #byte	3	3	—	$\text{PSW} \leftarrow \text{byte}$	x	x	x
		CS, #byte	3	1	—	$\text{CS} \leftarrow \text{byte}$			
		ES, #byte	2	1	—	$\text{ES} \leftarrow \text{byte}$			
		!addr16, #byte	4	1	—	$(\text{addr16}) \leftarrow \text{byte}$			
		ES:!addr16, #byte	5	2	—	$(\text{ES}, \text{addr16}) \leftarrow \text{byte}$			
		saddr, #byte	3	1	—	$(\text{saddr}) \leftarrow \text{byte}$			
		sfr, #byte	3	1	—	$\text{sfr} \leftarrow \text{byte}$			
		[DE+byte], #byte	3	1	—	$(\text{DE} + \text{byte}) \leftarrow \text{byte}$			
		ES:[DE+byte], #byte	4	2	—	$((\text{ES}, \text{DE}) + \text{byte}) \leftarrow \text{byte}$			
		[HL+byte], #byte	3	1	—	$(\text{HL} + \text{byte}) \leftarrow \text{byte}$			
		ES:[HL+byte], #byte	4	2	—	$((\text{ES}, \text{HL}) + \text{byte}) \leftarrow \text{byte}$			
		[SP+byte], #byte	3	1	—	$(\text{SP} + \text{byte}) \leftarrow \text{byte}$			
		word[B], #byte	4	1	—	$(\text{B} + \text{word}) \leftarrow \text{byte}$			
		ES:word[B], #byte	5	2	—	$((\text{ES}, \text{B}) + \text{word}) \leftarrow \text{byte}$			
		word[C], #byte	4	1	—	$(\text{C} + \text{word}) \leftarrow \text{byte}$			
		ES:word[C], #byte	5	2	—	$((\text{ES}, \text{C}) + \text{word}) \leftarrow \text{byte}$			
		word[BC], #byte	4	1	—	$(\text{BC} + \text{word}) \leftarrow \text{byte}$			
		ES:word[BC], #byte	5	2	—	$((\text{ES}, \text{BC}) + \text{word}) \leftarrow \text{byte}$			
		A, r ^{Note 3}	1	1	—	$\text{A} \leftarrow r$			
		r ^{Note 3} , A	1	1	—	$r \leftarrow \text{A}$			
		A, PSW	2	1	—	$\text{A} \leftarrow \text{PSW}$			
		PSW, A	2	3	—	$\text{PSW} \leftarrow \text{A}$	x	x	x
		A, CS	2	1	—	$\text{A} \leftarrow \text{CS}$			
		CS, A	2	1	—	$\text{CS} \leftarrow \text{A}$			
		A, ES	2	1	—	$\text{A} \leftarrow \text{ES}$			
		ES, A	2	1	—	$\text{ES} \leftarrow \text{A}$			
		A, !addr16	3	1	4	$\text{A} \leftarrow (\text{addr16})$			
		A, ES:!addr16	4	2	5	$\text{A} \leftarrow (\text{ES}, \text{addr16})$			
		!addr16, A	3	1	—	$(\text{addr16}) \leftarrow \text{A}$			
		ES:!addr16, A	4	2	—	$(\text{ES}, \text{addr16}) \leftarrow \text{A}$			
		A, saddr	2	1	—	$\text{A} \leftarrow (\text{saddr})$			
		saddr, A	2	1	—	$(\text{saddr}) \leftarrow \text{A}$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (2/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	A, sfr	2	1	—	$A \leftarrow \text{sfr}$			
		sfr, A	2	1	—	$\text{sfr} \leftarrow A$			
		A, [DE]	1	1	4	$A \leftarrow (DE)$			
		[DE], A	1	1	—	$(DE) \leftarrow A$			
		A, ES:[DE]	2	2	5	$A \leftarrow (ES, DE)$			
		ES:[DE], A	2	2	—	$(ES, DE) \leftarrow A$			
		A, [HL]	1	1	4	$A \leftarrow (HL)$			
		[HL], A	1	1	—	$(HL) \leftarrow A$			
		A, ES:[HL]	2	2	5	$A \leftarrow (ES, HL)$			
		ES:[HL], A	2	2	—	$(ES, HL) \leftarrow A$			
		A, [DE+byte]	2	1	4	$A \leftarrow (DE + \text{byte})$			
		[DE+byte], A	2	1	—	$(DE + \text{byte}) \leftarrow A$			
		A, ES:[DE+byte]	3	2	5	$A \leftarrow ((ES, DE) + \text{byte})$			
		ES:[DE+byte], A	3	2	—	$((ES, DE) + \text{byte}) \leftarrow A$			
		A, [HL+byte]	2	1	4	$A \leftarrow (HL + \text{byte})$			
		[HL+byte], A	2	1	—	$(HL + \text{byte}) \leftarrow A$			
		A, ES:[HL+byte]	3	2	5	$A \leftarrow ((ES, HL) + \text{byte})$			
		ES:[HL+byte], A	3	2	—	$((ES, HL) + \text{byte}) \leftarrow A$			
		A, [SP+byte]	2	1	—	$A \leftarrow (SP + \text{byte})$			
		[SP+byte], A	2	1	—	$(SP + \text{byte}) \leftarrow A$			
		A, word[B]	3	1	4	$A \leftarrow (B + \text{word})$			
		word[B], A	3	1	—	$(B + \text{word}) \leftarrow A$			
		A, ES:word[B]	4	2	5	$A \leftarrow ((ES, B) + \text{word})$			
		ES:word[B], A	4	2	—	$((ES, B) + \text{word}) \leftarrow A$			
		A, word[C]	3	1	4	$A \leftarrow (C + \text{word})$			
		word[C], A	3	1	—	$(C + \text{word}) \leftarrow A$			
		A, ES:word[C]	4	2	5	$A \leftarrow ((ES, C) + \text{word})$			
		ES:word[C], A	4	2	—	$((ES, C) + \text{word}) \leftarrow A$			
		A, word[BC]	3	1	4	$A \leftarrow (BC + \text{word})$			
		word[BC], A	3	1	—	$(BC + \text{word}) \leftarrow A$			
		A, ES:word[BC]	4	2	5	$A \leftarrow ((ES, BC) + \text{word})$			
		ES:word[BC], A	4	2	—	$((ES, BC) + \text{word}) \leftarrow A$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (3/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	A, [HL+B]	2	1	4	$A \leftarrow (HL + B)$			
		[HL+B], A	2	1	—	$(HL + B) \leftarrow A$			
		A, ES:[HL+B]	3	2	5	$A \leftarrow ((ES, HL) + B)$			
		ES:[HL+B], A	3	2	—	$((ES, HL) + B) \leftarrow A$			
		A, [HL+C]	2	1	4	$A \leftarrow (HL + C)$			
		[HL+C], A	2	1	—	$(HL + C) \leftarrow A$			
		A, ES:[HL+C]	3	2	5	$A \leftarrow ((ES, HL) + C)$			
		ES:[HL+C], A	3	2	—	$((ES, HL) + C) \leftarrow A$			
		X, !addr16	3	1	4	$X \leftarrow (addr16)$			
		X, ES:!addr16	4	2	5	$X \leftarrow (ES, addr16)$			
		X, saddr	2	1	—	$X \leftarrow (saddr)$			
		B, !addr16	3	1	4	$B \leftarrow (addr16)$			
		B, ES:!addr16	4	2	5	$B \leftarrow (ES, addr16)$			
		B, saddr	2	1	—	$B \leftarrow (saddr)$			
		C, !addr16	3	1	4	$C \leftarrow (addr16)$			
		C, ES:!addr16	4	2	5	$C \leftarrow (ES, addr16)$			
		C, saddr	2	1	—	$C \leftarrow (saddr)$			
		ES, saddr	3	1	—	$ES \leftarrow (saddr)$			
	XCH	A, r ^{Note 3}	1 (r = X) 2 (other than r = X)	1	—	$A \longleftrightarrow r$			
		A, !addr16	4	2	—	$A \longleftrightarrow (addr16)$			
		A, ES:!addr16	5	3	—	$A \longleftrightarrow (ES, addr16)$			
		A, saddr	3	2	—	$A \longleftrightarrow (saddr)$			
		A, sfr	3	2	—	$A \longleftrightarrow sfr$			
		A, [DE]	2	2	—	$A \longleftrightarrow (DE)$			
		A, ES:[DE]	3	3	—	$A \longleftrightarrow (ES, DE)$			
		A, [HL]	2	2	—	$A \longleftrightarrow (HL)$			
		A, ES:[HL]	3	3	—	$A \longleftrightarrow (ES, HL)$			
		A, [DE+byte]	3	2	—	$A \longleftrightarrow (DE + \text{byte})$			
		A, ES:[DE+byte]	4	3	—	$A \longleftrightarrow ((ES, DE) + \text{byte})$			
		A, [HL+byte]	3	2	—	$A \longleftrightarrow (HL + \text{byte})$			
		A, ES:[HL+byte]	4	3	—	$A \longleftrightarrow ((ES, HL) + \text{byte})$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (4/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	XCH	A, [HL+B]	2	2	—	$A \longleftrightarrow (HL + B)$			
		A, ES:[HL+B]	3	3	—	$A \longleftrightarrow ((ES, HL) + B)$			
		A, [HL+C]	2	2	—	$A \longleftrightarrow (HL + C)$			
		A, ES:[HL+C]	3	3	—	$A \longleftrightarrow ((ES, HL) + C)$			
	ONEB	A	1	1	—	$A \leftarrow 01H$			
		X	1	1	—	$X \leftarrow 01H$			
		B	1	1	—	$B \leftarrow 01H$			
		C	1	1	—	$C \leftarrow 01H$			
		!addr16	3	1	—	$(addr16) \leftarrow 01H$			
		ES:!addr16	4	2	—	$(ES, addr16) \leftarrow 01H$			
		saddr	2	1	—	$(saddr) \leftarrow 01H$			
	CLRB	A	1	1	—	$A \leftarrow 00H$			
		X	1	1	—	$X \leftarrow 00H$			
		B	1	1	—	$B \leftarrow 00H$			
		C	1	1	—	$C \leftarrow 00H$			
		!addr16	3	1	—	$(addr16) \leftarrow 00H$			
		ES:!addr16	4	2	—	$(ES, addr16) \leftarrow 00H$			
		saddr	2	1	—	$(saddr) \leftarrow 00H$			
	MOVS	[HL+byte], X	3	1	—	$(HL + byte) \leftarrow X$	x		x
		ES:[HL+byte], X	4	2	—	$(ES, HL + byte) \leftarrow X$	x		x
16-bit data transfer	MOVW	rp, #word	3	1	—	$rp \leftarrow word$			
		saddrp, #word	4	1	—	$(saddrp) \leftarrow word$			
		sfrp, #word	4	1	—	$sfrp \leftarrow word$			
		AX, rp ^{Note 3}	1	1	—	$AX \leftarrow rp$			
		rp ^{Note 3} , AX	1	1	—	$rp \leftarrow AX$			
		AX, !addr16	3	1	4	$AX \leftarrow (addr16)$			
		!addr16, AX	3	1	—	$(addr16) \leftarrow AX$			
		AX, ES:!addr16	4	2	5	$AX \leftarrow (ES, addr16)$			
		ES:!addr16, AX	4	2	—	$(ES, addr16) \leftarrow AX$			
		AX, saddrp	2	1	—	$AX \leftarrow (saddrp)$			
		saddrp, AX	2	1	—	$(saddrp) \leftarrow AX$			
		AX, sfrp	2	1	—	$AX \leftarrow sfrp$			
		sfrp, AX	2	1	—	$sfrp \leftarrow AX$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register pair other than AX.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (5/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit data transfer	MOVW	AX, [DE]	1	1	4	$AX \leftarrow (DE)$			
		[DE], AX	1	1	—	$(DE) \leftarrow AX$			
		AX, ES:[DE]	2	2	5	$AX \leftarrow (ES, DE)$			
		ES:[DE], AX	2	2	—	$(ES, DE) \leftarrow AX$			
		AX, [HL]	1	1	4	$AX \leftarrow (HL)$			
		[HL], AX	1	1	—	$(HL) \leftarrow AX$			
		AX, ES:[HL]	2	2	5	$AX \leftarrow (ES, HL)$			
		ES:[HL], AX	2	2	—	$(ES, HL) \leftarrow AX$			
		AX, [DE+byte]	2	1	4	$AX \leftarrow (DE + \text{byte})$			
		[DE+byte], AX	2	1	—	$(DE + \text{byte}) \leftarrow AX$			
		AX, ES:[DE+byte]	3	2	5	$AX \leftarrow ((ES, DE) + \text{byte})$			
		ES:[DE+byte], AX	3	2	—	$((ES, DE) + \text{byte}) \leftarrow AX$			
		AX, [HL+byte]	2	1	4	$AX \leftarrow (HL + \text{byte})$			
		[HL+byte], AX	2	1	—	$(HL + \text{byte}) \leftarrow AX$			
		AX, ES:[HL+byte]	3	2	5	$AX \leftarrow ((ES, HL) + \text{byte})$			
		ES:[HL+byte], AX	3	2	—	$((ES, HL) + \text{byte}) \leftarrow AX$			
		AX, [SP+byte]	2	1	—	$AX \leftarrow (SP + \text{byte})$			
		[SP+byte], AX	2	1	—	$(SP + \text{byte}) \leftarrow AX$			
		AX, word[B]	3	1	4	$AX \leftarrow (B + \text{word})$			
		word[B], AX	3	1	—	$(B + \text{word}) \leftarrow AX$			
		AX, ES:word[B]	4	2	5	$AX \leftarrow ((ES, B) + \text{word})$			
		ES:word[B], AX	4	2	—	$((ES, B) + \text{word}) \leftarrow AX$			
		AX, word[C]	3	1	4	$AX \leftarrow (C + \text{word})$			
		word[C], AX	3	1	—	$(C + \text{word}) \leftarrow AX$			
		AX, ES:word[C]	4	2	5	$AX \leftarrow ((ES, C) + \text{word})$			
		ES:word[C], AX	4	2	—	$((ES, C) + \text{word}) \leftarrow AX$			
		AX, word[BC]	3	1	4	$AX \leftarrow (BC + \text{word})$			
		word[BC], AX	3	1	—	$(BC + \text{word}) \leftarrow AX$			
		AX, ES:word[BC]	4	2	5	$AX \leftarrow ((ES, BC) + \text{word})$			
		ES:word[BC], AX	4	2	—	$((ES, BC) + \text{word}) \leftarrow AX$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (6/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit data transfer	MOVW	BC, !addr16	3	1	4	BC ← (addr16)			
		BC, ES:!addr16	4	2	5	BC ← (ES, addr16)			
		DE, !addr16	3	1	4	DE ← (addr16)			
		DE, ES:!addr16	4	2	5	DE ← (ES, addr16)			
		HL, !addr16	3	1	4	HL ← (addr16)			
		HL, ES:!addr16	4	2	5	HL ← (ES, addr16)			
		BC, saddrp	2	1	—	BC ← (saddrp)			
		DE, saddrp	2	1	—	DE ← (saddrp)			
		HL, saddrp	2	1	—	HL ← (saddrp)			
	XCHW	AX, rp ^{Note 3}	1	1	—	AX ↔ rp			
	ONEW	AX	1	1	—	AX ← 0001H			
		BC	1	1	—	BC ← 0001H			
	CLRW	AX	1	1	—	AX ← 0000H			
		BC	1	1	—	BC ← 0000H			
8-bit operation	ADD	A, #byte	2	1	—	A, CY ← A + byte	x	x	x
		saddr, #byte	3	2	—	(saddr), CY ← (saddr) + byte	x	x	x
		A, r ^{Note 4}	2	1	—	A, CY ← A + r	x	x	x
		r, A	2	1	—	r, CY ← r + A	x	x	x
		A, !addr16	3	1	4	A, CY ← A + (addr16)	x	x	x
		A, ES:!addr16	4	2	5	A, CY ← A + (ES, addr16)	x	x	x
		A, saddr	2	1	—	A, CY ← A + (saddr)	x	x	x
		A, [HL]	1	1	4	A, CY ← A + (HL)	x	x	x
		A, ES:[HL]	2	2	5	A, CY ← A + (ES, HL)	x	x	x
		A, [HL+byte]	2	1	4	A, CY ← A + (HL + byte)	x	x	x
		A, ES:[HL+byte]	3	2	5	A, CY ← A + ((ES, HL) + byte)	x	x	x
		A, [HL+B]	2	1	4	A, CY ← A + (HL + B)	x	x	x
		A, ES:[HL+B]	3	2	5	A, CY ← A + ((ES, HL) + B)	x	x	x
		A, [HL+C]	2	1	4	A, CY ← A + (HL + C)	x	x	x
		A, ES:[HL+C]	3	2	5	A, CY ← A + ((ES, HL) + C)	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register pair other than AX.

Note 4. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (7/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	ADDC	A, #byte	2	1	—	$A, CY \leftarrow A + \text{byte} + CY$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) + \text{byte} + CY$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A + r + CY$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r + A + CY$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A + (\text{addr16}) + CY$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A + (ES, \text{addr16}) + CY$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A + (saddr) + CY$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A + (HL) + CY$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A + (ES, HL) + CY$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A + (HL + \text{byte}) + CY$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + \text{byte}) + CY$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A + (HL + B) + CY$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + B) + CY$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A + (HL + C) + CY$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + C) + CY$	x	x	x
	SUB	A, #byte	2	1	—	$A, CY \leftarrow A - \text{byte}$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) - \text{byte}$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A - r$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r - A$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A - (\text{addr16})$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A - (ES, \text{addr16})$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A - (saddr)$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A - (HL)$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A - (ES, HL)$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A - (HL + \text{byte})$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + \text{byte})$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A - (HL + B)$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + B)$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A - (HL + C)$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + C)$	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (8/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	SUBC	A, #byte	2	1	—	$A, CY \leftarrow A - \text{byte} - CY$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) - \text{byte} - CY$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A - r - CY$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r - A - CY$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A - (\text{addr16}) - CY$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A - (ES, \text{addr16}) - CY$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A - (saddr) - CY$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A - (HL) - CY$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A - (ES, HL) - CY$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A - (HL + \text{byte}) - CY$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + \text{byte}) - CY$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A - (HL + B) - CY$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + B) - CY$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A - (HL + C) - CY$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A - ((ES:HL) + C) - CY$	x	x	x
	AND	A, #byte	2	1	—	$A \leftarrow A \wedge \text{byte}$	x		
		saddr, #byte	3	2	—	$(saddr) \leftarrow (saddr) \wedge \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \wedge r$	x		
		r, A	2	1	—	$r \leftarrow r \wedge A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \wedge (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \wedge (ES:\text{addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \wedge (saddr)$	x		
		A, [HL]	1	1	4	$A \leftarrow A \wedge (HL)$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \wedge (ES:HL)$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \wedge (HL + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \wedge ((ES:HL) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \wedge (HL + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \wedge ((ES:HL) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \wedge (HL + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \wedge ((ES:HL) + C)$	x		

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (9/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	OR	A, #byte	2	1	—	$A \leftarrow A \vee \text{byte}$	x		
		saddr, #byte	3	2	—	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \vee r$	x		
		r, A	2	1	—	$r \leftarrow r \vee A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \vee (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \vee (\text{ES:addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \vee (\text{saddr})$	x		
		A, [HL]	1	1	4	$A \leftarrow A \vee (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \vee (\text{ES:HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \vee (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \vee (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \vee (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + C)$	x		
	XOR	A, #byte	2	1	—	$A \leftarrow A \veebar \text{byte}$	x		
		saddr, #byte	3	2	—	$(\text{saddr}) \leftarrow (\text{saddr}) \veebar \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \veebar r$	x		
		r, A	2	1	—	$r \leftarrow r \veebar A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \veebar (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \veebar (\text{ES:addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \veebar (\text{saddr})$	x		
		A, [HL]	1	1	4	$A \leftarrow A \veebar (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \veebar (\text{ES:HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \veebar (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \veebar ((\text{ES:HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \veebar (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \veebar ((\text{ES:HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \veebar (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \veebar ((\text{ES:HL}) + C)$	x		

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (10/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	CMP	A, #byte	2	1	—	A - byte	x	x	x
		!addr16, #byte	4	1	4	(addr16) - byte	x	x	x
		ES:!addr16, #byte	5	2	5	(ES:addr16) - byte	x	x	x
		saddr, #byte	3	1	—	(saddr) - byte	x	x	x
		A, r ^{Note 3}	2	1	—	A - r	x	x	x
		r, A	2	1	—	r - A	x	x	x
		A, !addr16	3	1	4	A - (addr16)	x	x	x
		A, ES:!addr16	4	2	5	A - (ES:addr16)	x	x	x
		A, saddr	2	1	—	A - (saddr)	x	x	x
		A, [HL]	1	1	4	A - (HL)	x	x	x
		A, ES:[HL]	2	2	5	A - (ES:HL)	x	x	x
		A, [HL+byte]	2	1	4	A - (HL + byte)	x	x	x
		A, ES:[HL+byte]	3	2	5	A - ((ES:HL) + byte)	x	x	x
		A, [HL+B]	2	1	4	A - (HL + B)	x	x	x
		A, ES:[HL+B]	3	2	5	A - ((ES:HL) + B)	x	x	x
		A, [HL+C]	2	1	4	A - (HL + C)	x	x	x
		A, ES:[HL+C]	3	2	5	A - ((ES:HL) + C)	x	x	x
	CMP0	A	1	1	—	A - 00H	x	0	0
		X	1	1	—	X - 00H	x	0	0
		B	1	1	—	B - 00H	x	0	0
		C	1	1	—	C - 00H	x	0	0
		!addr16	3	1	4	(addr16) - 00H	x	0	0
		ES:!addr16	4	2	5	(ES:addr16) - 00H	x	0	0
		saddr	2	1	—	(saddr) - 00H	x	0	0
	CMPS	X, [HL+byte]	3	1	4	X - (HL + byte)	x	x	x
		X, ES:[HL+byte]	4	2	5	X - ((ES:HL) + byte)	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (11/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit operation	ADDW	AX, #word	3	1	—	AX, CY $\leftarrow$ AX + word	x	x	x
		AX, AX	1	1	—	AX, CY $\leftarrow$ AX + AX	x	x	x
		AX, BC	1	1	—	AX, CY $\leftarrow$ AX + BC	x	x	x
		AX, DE	1	1	—	AX, CY $\leftarrow$ AX + DE	x	x	x
		AX, HL	1	1	—	AX, CY $\leftarrow$ AX + HL	x	x	x
		AX, !addr16	3	1	4	AX, CY $\leftarrow$ AX + (addr16)	x	x	x
		AX, ES:!addr16	4	2	5	AX, CY $\leftarrow$ AX + (ES:addr16)	x	x	x
		AX, saddrp	2	1	—	AX, CY $\leftarrow$ AX + (saddrp)	x	x	x
		AX, [HL+byte]	3	1	4	AX, CY $\leftarrow$ AX + (HL + byte)	x	x	x
		AX, ES:[HL+byte]	4	2	5	AX, CY $\leftarrow$ AX + ((ES:HL) + byte)	x	x	x
	SUBW	AX, #word	3	1	—	AX, CY $\leftarrow$ AX - word	x	x	x
		AX, BC	1	1	—	AX, CY $\leftarrow$ AX - BC	x	x	x
		AX, DE	1	1	—	AX, CY $\leftarrow$ AX - DE	x	x	x
		AX, HL	1	1	—	AX, CY $\leftarrow$ AX - HL	x	x	x
		AX, !addr16	3	1	4	AX, CY $\leftarrow$ AX - (addr16)	x	x	x
		AX, ES:!addr16	4	2	5	AX, CY $\leftarrow$ AX - (ES:addr16)	x	x	x
		AX, saddrp	2	1	—	AX, CY $\leftarrow$ AX - (saddrp)	x	x	x
		AX, [HL+byte]	3	1	4	AX, CY $\leftarrow$ AX - (HL + byte)	x	x	x
		AX, ES:[HL+byte]	4	2	5	AX, CY $\leftarrow$ AX - ((ES:HL) + byte)	x	x	x
	CMPW	AX, #word	3	1	—	AX - word	x	x	x
		AX, BC	1	1	—	AX - BC	x	x	x
		AX, DE	1	1	—	AX - DE	x	x	x
		AX, HL	1	1	—	AX - HL	x	x	x
		AX, !addr16	3	1	4	AX - (addr16)	x	x	x
		AX, ES:!addr16	4	2	5	AX - (ES:addr16)	x	x	x
		AX, saddrp	2	1	—	AX - (saddrp)	x	x	x
		AX, [HL+byte]	3	1	4	AX - (HL + byte)	x	x	x
		AX, ES:[HL+byte]	4	2	5	AX - ((ES:HL) + byte)	x	x	x
Multiply	MULU	X	1	1	—	AX $\leftarrow$ A $\times$ X			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (12/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Increment and decrement	INC	r	1	1	—	$r \leftarrow r + 1$	x	x	
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) + 1$	x	x	
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) + 1$	x	x	
		saddr	2	2	—	$(saddr) \leftarrow (saddr) + 1$	x	x	
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) + 1$	x	x	
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) + 1$	x	x	
	DEC	r	1	1	—	$r \leftarrow r - 1$	x	x	
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) - 1$	x	x	
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) - 1$	x	x	
		saddr	2	2	—	$(saddr) \leftarrow (saddr) - 1$	x	x	
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) - 1$	x	x	
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) - 1$	x	x	
	INCW	rp	1	1	—	$rp \leftarrow rp + 1$			
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) + 1$			
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) + 1$			
		saddrp	2	2	—	$(saddrp) \leftarrow (saddrp) + 1$			
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) + 1$			
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) + 1$			
	DECW	rp	1	1	—	$rp \leftarrow rp - 1$			
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) - 1$			
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) - 1$			
		saddrp	2	2	—	$(saddrp) \leftarrow (saddrp) - 1$			
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) - 1$			
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) - 1$			
Shift	SHR	A, cnt	2	1	—	$(CY \leftarrow A_0, A_{m-1} \leftarrow A_m, A_7 \leftarrow 0) \times cnt$			x
	SHRW	AX, cnt	2	1	—	$(CY \leftarrow AX_0, AX_{m-1} \leftarrow AX_m, AX_{15} \leftarrow 0) \times cnt$			x
	SHL	A, cnt	2	1	—	$(CY \leftarrow A_7, A_m \leftarrow A_{m-1}, A_0 \leftarrow 0) \times cnt$			x
		B, cnt	2	1	—	$(CY \leftarrow B_7, B_m \leftarrow B_{m-1}, B_0 \leftarrow 0) \times cnt$			x
		C, cnt	2	1	—	$(CY \leftarrow C_7, C_m \leftarrow C_{m-1}, C_0 \leftarrow 0) \times cnt$			x
	SHLW	AX, cnt	2	1	—	$(CY \leftarrow AX_{15}, AX_m \leftarrow AX_{m-1}, AX_0 \leftarrow 0) \times cnt$			x
		BC, cnt	2	1	—	$(CY \leftarrow BC_{15}, BC_m \leftarrow BC_{m-1}, BC_0 \leftarrow 0) \times cnt$			x
	SAR	A, cnt	2	1	—	$(CY \leftarrow A_0, A_{m-1} \leftarrow A_m, A_7 \leftarrow A_7) \times cnt$			x
	SARW	AX, cnt	2	1	—	$(CY \leftarrow AX_0, AX_{m-1} \leftarrow AX_m, AX_{15} \leftarrow AX_{15}) \times cnt$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark 1. The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Remark 2. cnt indicates the bit shift count.

Table 5 - 6 Operation List of RL78-S2 Core (13/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Rotate	ROR	A, 1	2	1	—	$(CY, A7 \leftarrow A0, A_{m-1} \leftarrow A_m) \times 1$			x
	ROL	A, 1	2	1	—	$(CY, A0 \leftarrow A7, A_{m+1} \leftarrow A_m) \times 1$			x
	RORC	A, 1	2	1	—	$(CY \leftarrow A0, A7 \leftarrow CY, A_{m-1} \leftarrow A_m) \times 1$			x
	ROLC	A, 1	2	1	—	$(CY \leftarrow A7, A0 \leftarrow CY, A_{m+1} \leftarrow A_m) \times 1$			x
	ROLWC	AX, 1	2	1	—	$(CY \leftarrow AX_{15}, AX0 \leftarrow CY, AX_{m+1} \leftarrow AX_m) \times 1$			x
		BC, 1	2	1	—	$(CY \leftarrow BC_{15}, BC0 \leftarrow CY, BC_{m+1} \leftarrow BC_m) \times 1$			x
Bit manipulation	MOV1	CY, A.bit	2	1	—	$CY \leftarrow A.bit$			x
		A.bit, CY	2	1	—	$A.bit \leftarrow CY$			
		CY, PSW.bit	3	1	—	$CY \leftarrow PSW.bit$			x
		PSW.bit, CY	3	4	—	$PSW.bit \leftarrow CY$	x	x	
		CY, saddr.bit	3	1	—	$CY \leftarrow (saddr).bit$			x
		saddr.bit, CY	3	2	—	$(saddr).bit \leftarrow CY$			
		CY, sfr.bit	3	1	—	$CY \leftarrow sfr.bit$			x
		sfr.bit, CY	3	2	—	$sfr.bit \leftarrow CY$			
		CY, [HL].bit	2	1	4	$CY \leftarrow (HL).bit$			x
		[HL].bit, CY	2	2	—	$(HL).bit \leftarrow CY$			
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow (ES, HL).bit$			x
		ES:[HL].bit, CY	3	3	—	$(ES, HL).bit \leftarrow CY$			
	AND1	CY, A.bit	2	1	—	$CY \leftarrow CY \wedge A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \wedge PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \wedge (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \wedge sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \wedge (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \wedge (ES, HL).bit$			x
	OR1	CY, A.bit	2	1	—	$CY \leftarrow CY \vee A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \vee PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \vee (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \vee sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \vee (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \vee (ES, HL).bit$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (14/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Bit manipulation	XOR1	CY, A.bit	2	1	—	$CY \leftarrow CY \vee A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \vee PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \vee (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \vee sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \vee (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \vee (ES, HL).bit$			x
	SET1	A.bit	2	1	—	$A.bit \leftarrow 1$			
		PSW.bit	3	4	—	$PSW.bit \leftarrow 1$	x	x	x
		!addr16.bit	4	2	—	$(addr16).bit \leftarrow 1$			
		ES:!addr16.bit	5	3	—	$(ES, addr16).bit \leftarrow 1$			
		saddr.bit	3	2	—	$(saddr).bit \leftarrow 1$			
		sfr.bit	3	2	—	$sfr.bit \leftarrow 1$			
		[HL].bit	2	2	—	$(HL).bit \leftarrow 1$			
		ES:[HL].bit	3	3	—	$(ES, HL).bit \leftarrow 1$			
	CLR1	A.bit	2	1	—	$A.bit \leftarrow 0$			
		PSW.bit	3	4	—	$PSW.bit \leftarrow 0$	x	x	x
		!addr16.bit	4	2	—	$(addr16).bit \leftarrow 0$			
		ES:!addr16.bit	5	3	—	$(ES, addr16).bit \leftarrow 0$			
		saddr.bit	3	2	—	$(saddr).bit \leftarrow 0$			
		sfr.bit	3	2	—	$sfr.bit \leftarrow 0$			
		[HL].bit	2	2	—	$(HL).bit \leftarrow 0$			
		ES:[HL].bit	3	3	—	$(ES, HL).bit \leftarrow 0$			
	SET1	CY	2	1	—	$CY \leftarrow 1$			1
	CLR1	CY	2	1	—	$CY \leftarrow 0$			0
	NOT1	CY	2	1	—	$CY \leftarrow \overline{CY}$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (15/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Call and return	CALL	rp	2	3	—	$(SP - 2) \leftarrow (PC + 2)_S, (SP - 3) \leftarrow (PC + 2)_H,$ $(SP - 4) \leftarrow (PC + 2)_L, PC \leftarrow CS, rp,$ $SP \leftarrow SP - 4$			
		\$!addr20	3	3	—	$(SP - 2) \leftarrow (PC + 3)_S, (SP - 3) \leftarrow (PC + 3)_H,$ $(SP - 4) \leftarrow (PC + 3)_L, PC \leftarrow PC + 3 + jdisp16,$ $SP \leftarrow SP - 4$			
		!addr16	3	3	—	$(SP - 2) \leftarrow (PC + 3)_S, (SP - 3) \leftarrow (PC + 3)_H,$ $(SP - 4) \leftarrow (PC + 3)_L, PC \leftarrow 0000, addr16,$ $SP \leftarrow SP - 4$			
		!!addr20	4	3	—	$(SP - 2) \leftarrow (PC + 4)_S, (SP - 3) \leftarrow (PC + 4)_H,$ $(SP - 4) \leftarrow (PC + 4)_L, PC \leftarrow addr20,$ $SP \leftarrow SP - 4$			
	CALLT	[addr5]	2	5	—	$(SP - 2) \leftarrow (PC + 2)_S, (SP - 3) \leftarrow (PC + 2)_H,$ $(SP - 4) \leftarrow (PC + 2)_L, PCs \leftarrow 0000,$ $PCH \leftarrow (0000, addr5 + 1),$ $PCL \leftarrow (0000, addr5),$ $SP \leftarrow SP - 4$			
	BRK	—	2	5	—	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow (PC + 2)_S,$ $(SP - 3) \leftarrow (PC + 2)_H, (SP - 4) \leftarrow (PC + 2)_L,$ $PCs \leftarrow 0000,$ $PCH \leftarrow (0007FH), PCL \leftarrow (0007EH),$ $SP \leftarrow SP - 4, IE \leftarrow 0$			
	RET	—	1	6	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), SP \leftarrow SP + 4$			
	RETI	—	2	6	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), PSW \leftarrow (SP + 3),$ $SP \leftarrow SP + 4$	R	R	R
	RETB	—	2	6	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), PSW \leftarrow (SP + 3),$ $SP \leftarrow SP + 4$	R	R	R

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (16/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Stack manipulation	PUSH	PSW	2	1	—	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow 00H, SP \leftarrow SP - 2$			
		rp	1	1	—	$(SP - 1) \leftarrow rpH, (SP - 2) \leftarrow rpL, SP \leftarrow SP - 2$			
	POP	PSW	2	3	—	$PSW \leftarrow (SP + 1), SP \leftarrow SP + 2$	R	R	R
		rp	1	1	—	$rpL \leftarrow (SP), rpH \leftarrow (SP + 1), SP \leftarrow SP + 2$			
	MOVW	SP, #word	4	1	—	$SP \leftarrow word$			
		SP, AX	2	1	—	$SP \leftarrow AX$			
		AX, SP	2	1	—	$AX \leftarrow SP$			
		HL, SP	3	1	—	$HL \leftarrow SP$			
		BC, SP	3	1	—	$BC \leftarrow SP$			
		DE, SP	3	1	—	$DE \leftarrow SP$			
	ADDW	SP, #byte	2	1	—	$SP \leftarrow SP + byte$			
	SUBW	SP, #byte	2	1	—	$SP \leftarrow SP - byte$			
Unconditional branch	BR	AX	2	3	—	$PC \leftarrow CS, AX$			
		\$addr20	2	3	—	$PC \leftarrow PC + 2 + jdisp8$			
		\$!addr20	3	3	—	$PC \leftarrow PC + 3 + jdisp16$			
		!addr16	3	3	—	$PC \leftarrow 0000, addr16$			
		!addr20	4	3	—	$PC \leftarrow addr20$			
Conditional branch	BC	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 1$			
	BNC	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 0$			
	BZ	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 1$			
	BNZ	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 0$			
	BH	\$addr20	3	2/4 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if $(Z \vee CY) = 0$			
	BNH	\$addr20	3	2/4 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if $(Z \vee CY) = 1$			
	BT	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 1			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 1			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 1			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 1			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. These indicate the numbers of clock cycles when the condition is not and is met.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 6 Operation List of RL78-S2 Core (17/17)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Conditional branch	BT	[HL].bit, \$addr20	3	3/5 Note 3	6/7	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 1			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	7/8	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 1			
	BF	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 0			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 0			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 0			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 0			
		[HL].bit, \$addr20	3	3/5 Note 3	6/7	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 0			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	7/8	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 0			
	BTCLR	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 1 then reset (saddr).bit			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 1 then reset sfr.bit			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 1 then reset A.bit			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 1 then reset PSW.bit	x	x	x
		[HL].bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 1 then reset (HL).bit			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 1 then reset (ES, HL).bit			
Conditional skip	SKC	—	2	1	—	Next instruction skip if CY = 1			
	SKNC	—	2	1	—	Next instruction skip if CY = 0			
	SKZ	—	2	1	—	Next instruction skip if Z = 1			
	SKNZ	—	2	1	—	Next instruction skip if Z = 0			
	SKH	—	2	1	—	Next instruction skip if $(Z \vee CY) = 0$			
	SKNH	—	2	1	—	Next instruction skip if $(Z \vee CY) = 1$			
CPU control	SEL ^{Note 4}	RBn	2	1	—	$RBS[1:0] \leftarrow n$			
	NOP	—	1	1	—	No Operation			
	EI	—	3	4	—	$IE \leftarrow 1$ (Enable Interrupt)			
	DI	—	3	4	—	$IE \leftarrow 0$ (Disable Interrupt)			
	HALT	—	2	3	—	Set HALT Mode			
	STOP	—	2	3	—	Set STOP Mode			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. These indicate the numbers of clock cycles when the condition is not and is met.

Note 4. n indicates the number of register banks (n = 0 to 3).

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

5.5.3 Operation list of RL78-S3 core

Table 5 - 7 Operation List of RL78-S3 Core (1/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	r, #byte	2	1	—	$r \leftarrow \text{byte}$			
		PSW, #byte	3	3	—	$\text{PSW} \leftarrow \text{byte}$	x	x	x
		CS, #byte	3	1	—	$\text{CS} \leftarrow \text{byte}$			
		ES, #byte	2	1	—	$\text{ES} \leftarrow \text{byte}$			
		!addr16, #byte	4	1	—	$(\text{addr16}) \leftarrow \text{byte}$			
		ES:!addr16, #byte	5	2	—	$(\text{ES}, \text{addr16}) \leftarrow \text{byte}$			
		saddr, #byte	3	1	—	$(\text{saddr}) \leftarrow \text{byte}$			
		sfr, #byte	3	1	—	$\text{sfr} \leftarrow \text{byte}$			
		[DE+byte], #byte	3	1	—	$(\text{DE} + \text{byte}) \leftarrow \text{byte}$			
		ES:[DE+byte], #byte	4	2	—	$((\text{ES}, \text{DE}) + \text{byte}) \leftarrow \text{byte}$			
		[HL+byte], #byte	3	1	—	$(\text{HL} + \text{byte}) \leftarrow \text{byte}$			
		ES:[HL+byte], #byte	4	2	—	$((\text{ES}, \text{HL}) + \text{byte}) \leftarrow \text{byte}$			
		[SP+byte], #byte	3	1	—	$(\text{SP} + \text{byte}) \leftarrow \text{byte}$			
		word[B], #byte	4	1	—	$(\text{B} + \text{word}) \leftarrow \text{byte}$			
		ES:word[B], #byte	5	2	—	$((\text{ES}, \text{B}) + \text{word}) \leftarrow \text{byte}$			
		word[C], #byte	4	1	—	$(\text{C} + \text{word}) \leftarrow \text{byte}$			
		ES:word[C], #byte	5	2	—	$((\text{ES}, \text{C}) + \text{word}) \leftarrow \text{byte}$			
		word[BC], #byte	4	1	—	$(\text{BC} + \text{word}) \leftarrow \text{byte}$			
		ES:word[BC], #byte	5	2	—	$((\text{ES}, \text{BC}) + \text{word}) \leftarrow \text{byte}$			
		A, r ^{Note 3}	1	1	—	$\text{A} \leftarrow r$			
		r ^{Note 3} , A	1	1	—	$r \leftarrow \text{A}$			
		A, PSW	2	1	—	$\text{A} \leftarrow \text{PSW}$			
		PSW, A	2	3	—	$\text{PSW} \leftarrow \text{A}$	x	x	x
		A, CS	2	1	—	$\text{A} \leftarrow \text{CS}$			
		CS, A	2	1	—	$\text{CS} \leftarrow \text{A}$			
		A, ES	2	1	—	$\text{A} \leftarrow \text{ES}$			
		ES, A	2	1	—	$\text{ES} \leftarrow \text{A}$			
		A, !addr16	3	1	4	$\text{A} \leftarrow (\text{addr16})$			
		A, ES:!addr16	4	2	5	$\text{A} \leftarrow (\text{ES}, \text{addr16})$			
		!addr16, A	3	1	—	$(\text{addr16}) \leftarrow \text{A}$			
		ES:!addr16, A	4	2	—	$(\text{ES}, \text{addr16}) \leftarrow \text{A}$			
		A, saddr	2	1	—	$\text{A} \leftarrow (\text{saddr})$			
		saddr, A	2	1	—	$(\text{saddr}) \leftarrow \text{A}$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (2/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	A, sfr	2	1	—	$A \leftarrow \text{sfr}$			
		sfr, A	2	1	—	$\text{sfr} \leftarrow A$			
		A, [DE]	1	1	4	$A \leftarrow (DE)$			
		[DE], A	1	1	—	$(DE) \leftarrow A$			
		A, ES:[DE]	2	2	5	$A \leftarrow (ES, DE)$			
		ES:[DE], A	2	2	—	$(ES, DE) \leftarrow A$			
		A, [HL]	1	1	4	$A \leftarrow (HL)$			
		[HL], A	1	1	—	$(HL) \leftarrow A$			
		A, ES:[HL]	2	2	5	$A \leftarrow (ES, HL)$			
		ES:[HL], A	2	2	—	$(ES, HL) \leftarrow A$			
		A, [DE+byte]	2	1	4	$A \leftarrow (DE + \text{byte})$			
		[DE+byte], A	2	1	—	$(DE + \text{byte}) \leftarrow A$			
		A, ES:[DE+byte]	3	2	5	$A \leftarrow ((ES, DE) + \text{byte})$			
		ES:[DE+byte], A	3	2	—	$((ES, DE) + \text{byte}) \leftarrow A$			
		A, [HL+byte]	2	1	4	$A \leftarrow (HL + \text{byte})$			
		[HL+byte], A	2	1	—	$(HL + \text{byte}) \leftarrow A$			
		A, ES:[HL+byte]	3	2	5	$A \leftarrow ((ES, HL) + \text{byte})$			
		ES:[HL+byte], A	3	2	—	$((ES, HL) + \text{byte}) \leftarrow A$			
		A, [SP+byte]	2	1	—	$A \leftarrow (SP + \text{byte})$			
		[SP+byte], A	2	1	—	$(SP + \text{byte}) \leftarrow A$			
		A, word[B]	3	1	4	$A \leftarrow (B + \text{word})$			
		word[B], A	3	1	—	$(B + \text{word}) \leftarrow A$			
		A, ES:word[B]	4	2	5	$A \leftarrow ((ES, B) + \text{word})$			
		ES:word[B], A	4	2	—	$((ES, B) + \text{word}) \leftarrow A$			
		A, word[C]	3	1	4	$A \leftarrow (C + \text{word})$			
		word[C], A	3	1	—	$(C + \text{word}) \leftarrow A$			
		A, ES:word[C]	4	2	5	$A \leftarrow ((ES, C) + \text{word})$			
		ES:word[C], A	4	2	—	$((ES, C) + \text{word}) \leftarrow A$			
		A, word[BC]	3	1	4	$A \leftarrow (BC + \text{word})$			
		word[BC], A	3	1	—	$(BC + \text{word}) \leftarrow A$			
		A, ES:word[BC]	4	2	5	$A \leftarrow ((ES, BC) + \text{word})$			
		ES:word[BC], A	4	2	—	$((ES, BC) + \text{word}) \leftarrow A$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (3/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	MOV	A, [HL+B]	2	1	4	$A \leftarrow (HL + B)$			
		[HL+B], A	2	1	—	$(HL + B) \leftarrow A$			
		A, ES:[HL+B]	3	2	5	$A \leftarrow ((ES, HL) + B)$			
		ES:[HL+B], A	3	2	—	$((ES, HL) + B) \leftarrow A$			
		A, [HL+C]	2	1	4	$A \leftarrow (HL + C)$			
		[HL+C], A	2	1	—	$(HL + C) \leftarrow A$			
		A, ES:[HL+C]	3	2	5	$A \leftarrow ((ES, HL) + C)$			
		ES:[HL+C], A	3	2	—	$((ES, HL) + C) \leftarrow A$			
		X, !addr16	3	1	4	$X \leftarrow (addr16)$			
		X, ES:!addr16	4	2	5	$X \leftarrow (ES, addr16)$			
		X, saddr	2	1	—	$X \leftarrow (saddr)$			
		B, !addr16	3	1	4	$B \leftarrow (addr16)$			
		B, ES:!addr16	4	2	5	$B \leftarrow (ES, addr16)$			
		B, saddr	2	1	—	$B \leftarrow (saddr)$			
		C, !addr16	3	1	4	$C \leftarrow (addr16)$			
		C, ES:!addr16	4	2	5	$C \leftarrow (ES, addr16)$			
		C, saddr	2	1	—	$C \leftarrow (saddr)$			
		ES, saddr	3	1	—	$ES \leftarrow (saddr)$			
	XCH	A, r ^{Note 3}	1 (r = X) 2 (other than r = X)	1	—	$A \longleftrightarrow r$			
		A, !addr16	4	2	—	$A \longleftrightarrow (addr16)$			
		A, ES:!addr16	5	3	—	$A \longleftrightarrow (ES, addr16)$			
		A, saddr	3	2	—	$A \longleftrightarrow (saddr)$			
		A, sfr	3	2	—	$A \longleftrightarrow sfr$			
		A, [DE]	2	2	—	$A \longleftrightarrow (DE)$			
		A, ES:[DE]	3	3	—	$A \longleftrightarrow (ES, DE)$			
		A, [HL]	2	2	—	$A \longleftrightarrow (HL)$			
		A, ES:[HL]	3	3	—	$A \longleftrightarrow (ES, HL)$			
		A, [DE+byte]	3	2	—	$A \longleftrightarrow (DE + \text{byte})$			
		A, ES:[DE+byte]	4	3	—	$A \longleftrightarrow ((ES, DE) + \text{byte})$			
		A, [HL+byte]	3	2	—	$A \longleftrightarrow (HL + \text{byte})$			
		A, ES:[HL+byte]	4	3	—	$A \longleftrightarrow ((ES, HL) + \text{byte})$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (4/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit data transfer	XCH	A, [HL+B]	2	2	—	$A \longleftrightarrow (HL + B)$			
		A, ES:[HL+B]	3	3	—	$A \longleftrightarrow ((ES, HL) + B)$			
		A, [HL+C]	2	2	—	$A \longleftrightarrow (HL + C)$			
		A, ES:[HL+C]	3	3	—	$A \longleftrightarrow ((ES, HL) + C)$			
	ONEB	A	1	1	—	$A \leftarrow 01H$			
		X	1	1	—	$X \leftarrow 01H$			
		B	1	1	—	$B \leftarrow 01H$			
		C	1	1	—	$C \leftarrow 01H$			
		!addr16	3	1	—	$(addr16) \leftarrow 01H$			
		ES:!addr16	4	2	—	$(ES, addr16) \leftarrow 01H$			
		saddr	2	1	—	$(saddr) \leftarrow 01H$			
	CLRB	A	1	1	—	$A \leftarrow 00H$			
		X	1	1	—	$X \leftarrow 00H$			
		B	1	1	—	$B \leftarrow 00H$			
		C	1	1	—	$C \leftarrow 00H$			
		!addr16	3	1	—	$(addr16) \leftarrow 00H$			
		ES:!addr16	4	2	—	$(ES, addr16) \leftarrow 00H$			
		saddr	2	1	—	$(saddr) \leftarrow 00H$			
	MOVS	[HL+byte], X	3	1	—	$(HL + byte) \leftarrow X$	x		x
		ES:[HL+byte], X	4	2	—	$(ES, HL + byte) \leftarrow X$	x		x
16-bit data transfer	MOVW	rp, #word	3	1	—	$rp \leftarrow word$			
		saddrp, #word	4	1	—	$(saddrp) \leftarrow word$			
		sfrp, #word	4	1	—	$sfrp \leftarrow word$			
		AX, rp ^{Note 3}	1	1	—	$AX \leftarrow rp$			
		rp ^{Note 3} , AX	1	1	—	$rp \leftarrow AX$			
		AX, !addr16	3	1	4	$AX \leftarrow (addr16)$			
		!addr16, AX	3	1	—	$(addr16) \leftarrow AX$			
		AX, ES:!addr16	4	2	5	$AX \leftarrow (ES, addr16)$			
		ES:!addr16, AX	4	2	—	$(ES, addr16) \leftarrow AX$			
		AX, saddrp	2	1	—	$AX \leftarrow (saddrp)$			
		saddrp, AX	2	1	—	$(saddrp) \leftarrow AX$			
		AX, sfrp	2	1	—	$AX \leftarrow sfrp$			
		sfrp, AX	2	1	—	$sfrp \leftarrow AX$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register pair other than AX.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (5/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit data transfer	MOVW	AX, [DE]	1	1	4	$AX \leftarrow (DE)$			
		[DE], AX	1	1	—	$(DE) \leftarrow AX$			
		AX, ES:[DE]	2	2	5	$AX \leftarrow (ES, DE)$			
		ES:[DE], AX	2	2	—	$(ES, DE) \leftarrow AX$			
		AX, [HL]	1	1	4	$AX \leftarrow (HL)$			
		[HL], AX	1	1	—	$(HL) \leftarrow AX$			
		AX, ES:[HL]	2	2	5	$AX \leftarrow (ES, HL)$			
		ES:[HL], AX	2	2	—	$(ES, HL) \leftarrow AX$			
		AX, [DE+byte]	2	1	4	$AX \leftarrow (DE + \text{byte})$			
		[DE+byte], AX	2	1	—	$(DE + \text{byte}) \leftarrow AX$			
		AX, ES:[DE+byte]	3	2	5	$AX \leftarrow ((ES, DE) + \text{byte})$			
		ES:[DE+byte], AX	3	2	—	$((ES, DE) + \text{byte}) \leftarrow AX$			
		AX, [HL+byte]	2	1	4	$AX \leftarrow (HL + \text{byte})$			
		[HL+byte], AX	2	1	—	$(HL + \text{byte}) \leftarrow AX$			
		AX, ES:[HL+byte]	3	2	5	$AX \leftarrow ((ES, HL) + \text{byte})$			
		ES:[HL+byte], AX	3	2	—	$((ES, HL) + \text{byte}) \leftarrow AX$			
		AX, [SP+byte]	2	1	—	$AX \leftarrow (SP + \text{byte})$			
		[SP+byte], AX	2	1	—	$(SP + \text{byte}) \leftarrow AX$			
		AX, word[B]	3	1	4	$AX \leftarrow (B + \text{word})$			
		word[B], AX	3	1	—	$(B + \text{word}) \leftarrow AX$			
		AX, ES:word[B]	4	2	5	$AX \leftarrow ((ES, B) + \text{word})$			
		ES:word[B], AX	4	2	—	$((ES, B) + \text{word}) \leftarrow AX$			
		AX, word[C]	3	1	4	$AX \leftarrow (C + \text{word})$			
		word[C], AX	3	1	—	$(C + \text{word}) \leftarrow AX$			
		AX, ES:word[C]	4	2	5	$AX \leftarrow ((ES, C) + \text{word})$			
		ES:word[C], AX	4	2	—	$((ES, C) + \text{word}) \leftarrow AX$			
		AX, word[BC]	3	1	4	$AX \leftarrow (BC + \text{word})$			
		word[BC], AX	3	1	—	$(BC + \text{word}) \leftarrow AX$			
		AX, ES:word[BC]	4	2	5	$AX \leftarrow ((ES, BC) + \text{word})$			
		ES:word[BC], AX	4	2	—	$((ES, BC) + \text{word}) \leftarrow AX$			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (6/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit data transfer	MOVW	BC, !addr16	3	1	4	$BC \leftarrow (addr16)$			
		BC, ES:!addr16	4	2	5	$BC \leftarrow (ES, addr16)$			
		DE, !addr16	3	1	4	$DE \leftarrow (addr16)$			
		DE, ES:!addr16	4	2	5	$DE \leftarrow (ES, addr16)$			
		HL, !addr16	3	1	4	$HL \leftarrow (addr16)$			
		HL, ES:!addr16	4	2	5	$HL \leftarrow (ES, addr16)$			
		BC, saddrp	2	1	—	$BC \leftarrow (saddrp)$			
		DE, saddrp	2	1	—	$DE \leftarrow (saddrp)$			
		HL, saddrp	2	1	—	$HL \leftarrow (saddrp)$			
	XCHW	AX, rp ^{Note 3}	1	1	—	$AX \leftrightarrow rp$			
	ONEW	AX	1	1	—	$AX \leftarrow 0001H$			
		BC	1	1	—	$BC \leftarrow 0001H$			
	CLRW	AX	1	1	—	$AX \leftarrow 0000H$			
		BC	1	1	—	$BC \leftarrow 0000H$			
8-bit operation	ADD	A, #byte	2	1	—	$A, CY \leftarrow A + \text{byte}$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) + \text{byte}$	x	x	x
		A, r ^{Note 4}	2	1	—	$A, CY \leftarrow A + r$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r + A$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A + (addr16)$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A + (ES, addr16)$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A + (saddr)$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A + (HL)$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A + (ES, HL)$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A + (HL + \text{byte})$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + \text{byte})$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A + (HL + B)$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + B)$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A + (HL + C)$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + C)$	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register pair other than AX.

Note 4. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (7/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	ADDC	A, #byte	2	1	—	$A, CY \leftarrow A + \text{byte} + CY$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) + \text{byte} + CY$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A + r + CY$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r + A + CY$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A + (\text{addr16}) + CY$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A + (ES, \text{addr16}) + CY$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A + (saddr) + CY$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A + (HL) + CY$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A + (ES, HL) + CY$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A + (HL + \text{byte}) + CY$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + \text{byte}) + CY$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A + (HL + B) + CY$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + B) + CY$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A + (HL + C) + CY$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A + ((ES, HL) + C) + CY$	x	x	x
	SUB	A, #byte	2	1	—	$A, CY \leftarrow A - \text{byte}$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) - \text{byte}$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A - r$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r - A$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A - (\text{addr16})$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A - (ES, \text{addr16})$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A - (saddr)$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A - (HL)$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A - (ES, HL)$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A - (HL + \text{byte})$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + \text{byte})$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A - (HL + B)$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + B)$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A - (HL + C)$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A - ((ES, HL) + C)$	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (8/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	SUBC	A, #byte	2	1	—	$A, CY \leftarrow A - \text{byte} - CY$	x	x	x
		saddr, #byte	3	2	—	$(saddr), CY \leftarrow (saddr) - \text{byte} - CY$	x	x	x
		A, r ^{Note 3}	2	1	—	$A, CY \leftarrow A - r - CY$	x	x	x
		r, A	2	1	—	$r, CY \leftarrow r - A - CY$	x	x	x
		A, !addr16	3	1	4	$A, CY \leftarrow A - (\text{addr16}) - CY$	x	x	x
		A, ES:!addr16	4	2	5	$A, CY \leftarrow A - (\text{ES}, \text{addr16}) - CY$	x	x	x
		A, saddr	2	1	—	$A, CY \leftarrow A - (saddr) - CY$	x	x	x
		A, [HL]	1	1	4	$A, CY \leftarrow A - (\text{HL}) - CY$	x	x	x
		A, ES:[HL]	2	2	5	$A, CY \leftarrow A - (\text{ES}, \text{HL}) - CY$	x	x	x
		A, [HL+byte]	2	1	4	$A, CY \leftarrow A - (\text{HL} + \text{byte}) - CY$	x	x	x
		A, ES:[HL+byte]	3	2	5	$A, CY \leftarrow A - ((\text{ES}, \text{HL}) + \text{byte}) - CY$	x	x	x
		A, [HL+B]	2	1	4	$A, CY \leftarrow A - (\text{HL} + B) - CY$	x	x	x
		A, ES:[HL+B]	3	2	5	$A, CY \leftarrow A - ((\text{ES}, \text{HL}) + B) - CY$	x	x	x
		A, [HL+C]	2	1	4	$A, CY \leftarrow A - (\text{HL} + C) - CY$	x	x	x
		A, ES:[HL+C]	3	2	5	$A, CY \leftarrow A - ((\text{ES}, \text{HL}) + C) - CY$	x	x	x
	AND	A, #byte	2	1	—	$A \leftarrow A \wedge \text{byte}$	x		
		saddr, #byte	3	2	—	$(saddr) \leftarrow (saddr) \wedge \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \wedge r$	x		
		r, A	2	1	—	$r \leftarrow r \wedge A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \wedge (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \wedge (\text{ES}, \text{addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \wedge (saddr)$	x		
		A, [HL]	1	1	4	$A \leftarrow A \wedge (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \wedge (\text{ES}, \text{HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \wedge (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \wedge ((\text{ES}, \text{HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \wedge (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \wedge ((\text{ES}, \text{HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \wedge (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \wedge ((\text{ES}, \text{HL}) + C)$	x		

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (9/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	OR	A, #byte	2	1	—	$A \leftarrow A \vee \text{byte}$	x		
		saddr, #byte	3	2	—	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \vee r$	x		
		r, A	2	1	—	$r \leftarrow r \vee A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \vee (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \vee (\text{ES:addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \vee (\text{saddr})$	x		
		A, [HL]	1	1	4	$A \leftarrow A \vee (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \vee (\text{ES:HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \vee (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \vee (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \vee (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + C)$	x		
	XOR	A, #byte	2	1	—	$A \leftarrow A \vee \text{byte}$	x		
		saddr, #byte	3	2	—	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x		
		A, r ^{Note 3}	2	1	—	$A \leftarrow A \vee r$	x		
		r, A	2	1	—	$r \leftarrow r \vee A$	x		
		A, !addr16	3	1	4	$A \leftarrow A \vee (\text{addr16})$	x		
		A, ES:!addr16	4	2	5	$A \leftarrow A \vee (\text{ES:addr16})$	x		
		A, saddr	2	1	—	$A \leftarrow A \vee (\text{saddr})$	x		
		A, [HL]	1	1	4	$A \leftarrow A \vee (\text{HL})$	x		
		A, ES:[HL]	2	2	5	$A \leftarrow A \vee (\text{ES:HL})$	x		
		A, [HL+byte]	2	1	4	$A \leftarrow A \vee (\text{HL} + \text{byte})$	x		
		A, ES:[HL+byte]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + \text{byte})$	x		
		A, [HL+B]	2	1	4	$A \leftarrow A \vee (\text{HL} + B)$	x		
		A, ES:[HL+B]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + B)$	x		
		A, [HL+C]	2	1	4	$A \leftarrow A \vee (\text{HL} + C)$	x		
		A, ES:[HL+C]	3	2	5	$A \leftarrow A \vee ((\text{ES:HL}) + C)$	x		

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (10/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	CMP	A, #byte	2	1	—	A - byte	x	x	x
		!addr16, #byte	4	1	4	(addr16) - byte	x	x	x
		ES:!addr16, #byte	5	2	5	(ES:addr16) - byte	x	x	x
		saddr, #byte	3	1	—	(saddr) - byte	x	x	x
		A, r ^{Note 3}	2	1	—	A - r	x	x	x
		r, A	2	1	—	r - A	x	x	x
		A, !addr16	3	1	4	A - (addr16)	x	x	x
		A, ES:!addr16	4	2	5	A - (ES:addr16)	x	x	x
		A, saddr	2	1	—	A - (saddr)	x	x	x
		A, [HL]	1	1	4	A - (HL)	x	x	x
		A, ES:[HL]	2	2	5	A - (ES:HL)	x	x	x
		A, [HL+byte]	2	1	4	A - (HL + byte)	x	x	x
		A, ES:[HL+byte]	3	2	5	A - ((ES:HL) + byte)	x	x	x
		A, [HL+B]	2	1	4	A - (HL + B)	x	x	x
		A, ES:[HL+B]	3	2	5	A - ((ES:HL) + B)	x	x	x
		A, [HL+C]	2	1	4	A - (HL + C)	x	x	x
		A, ES:[HL+C]	3	2	5	A - ((ES:HL) + C)	x	x	x
	CMP0	A	1	1	—	A - 00H	x	0	0
		X	1	1	—	X - 00H	x	0	0
		B	1	1	—	B - 00H	x	0	0
		C	1	1	—	C - 00H	x	0	0
		!addr16	3	1	4	(addr16) - 00H	x	0	0
		ES:!addr16	4	2	5	(ES:addr16) - 00H	x	0	0
		saddr	2	1	—	(saddr) - 00H	x	0	0
	CMPS	X, [HL+byte]	3	1	4	X - (HL + byte)	x	x	x
		X, ES:[HL+byte]	4	2	5	X - ((ES:HL) + byte)	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. Specify a register other than A.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (11/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
16-bit operation	ADDW	AX, #word	3	1	—	$AX, CY \leftarrow AX + \text{word}$	x	x	x
		AX, AX	1	1	—	$AX, CY \leftarrow AX + AX$	x	x	x
		AX, BC	1	1	—	$AX, CY \leftarrow AX + BC$	x	x	x
		AX, DE	1	1	—	$AX, CY \leftarrow AX + DE$	x	x	x
		AX, HL	1	1	—	$AX, CY \leftarrow AX + HL$	x	x	x
		AX, !addr16	3	1	4	$AX, CY \leftarrow AX + (\text{addr16})$	x	x	x
		AX, ES:!addr16	4	2	5	$AX, CY \leftarrow AX + (\text{ES:addr16})$	x	x	x
		AX, saddrp	2	1	—	$AX, CY \leftarrow AX + (\text{saddrp})$	x	x	x
		AX, [HL+byte]	3	1	4	$AX, CY \leftarrow AX + (\text{HL} + \text{byte})$	x	x	x
		AX, ES:[HL+byte]	4	2	5	$AX, CY \leftarrow AX + ((\text{ES:HL}) + \text{byte})$	x	x	x
	SUBW	AX, #word	3	1	—	$AX, CY \leftarrow AX - \text{word}$	x	x	x
		AX, BC	1	1	—	$AX, CY \leftarrow AX - BC$	x	x	x
		AX, DE	1	1	—	$AX, CY \leftarrow AX - DE$	x	x	x
		AX, HL	1	1	—	$AX, CY \leftarrow AX - HL$	x	x	x
		AX, !addr16	3	1	4	$AX, CY \leftarrow AX - (\text{addr16})$	x	x	x
		AX, ES:!addr16	4	2	5	$AX, CY \leftarrow AX - (\text{ES:addr16})$	x	x	x
		AX, saddrp	2	1	—	$AX, CY \leftarrow AX - (\text{saddrp})$	x	x	x
		AX, [HL+byte]	3	1	4	$AX, CY \leftarrow AX - (\text{HL} + \text{byte})$	x	x	x
		AX, ES:[HL+byte]	4	2	5	$AX, CY \leftarrow AX - ((\text{ES:HL}) + \text{byte})$	x	x	x
	CMPW	AX, #word	3	1	—	$AX - \text{word}$	x	x	x
		AX, BC	1	1	—	$AX - BC$	x	x	x
		AX, DE	1	1	—	$AX - DE$	x	x	x
		AX, HL	1	1	—	$AX - HL$	x	x	x
		AX, !addr16	3	1	4	$AX - (\text{addr16})$	x	x	x
		AX, ES:!addr16	4	2	5	$AX - (\text{ES:addr16})$	x	x	x
		AX, saddrp	2	1	—	$AX - (\text{saddrp})$	x	x	x
		AX, [HL+byte]	3	1	4	$AX - (\text{HL} + \text{byte})$	x	x	x
		AX, ES:[HL+byte]	4	2	5	$AX - ((\text{ES:HL}) + \text{byte})$	x	x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (12/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Multiply, divide, and multiply-and-accumulate	MULU	X	1	1	—	$AX \leftarrow A \times X$			
	MULHU		3	2	—	$BCAX \leftarrow AX \times BC$ (unsigned)			
	MULH		3	2	—	$BCAX \leftarrow AX \times BC$ (signed)			
	DIVHU		3	9	—	AX (quotient), DE (remainder) $\leftarrow AX \div DE$ (unsigned)			
	DIVWU		3	17	—	$BCAX$ (quotient), $HLDE$ (remainder) $\leftarrow BCAX \div HLDE$ (unsigned)			
	MACHU		3	3	—	$MACR \leftarrow MACR + AX \times BC$ (unsigned)		x	x
	MACH		3	3	—	$MACR \leftarrow MACR + AX \times BC$ (signed)		x	x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

<R> **Caution** Disable interrupts when executing the DIVHU or DIVWU instruction in an interrupt servicing routine. Alternatively, unless they are executed in the RAM area, note that execution of a DIVHU or DIVWU instruction is possible even with interrupts enabled as long as a NOP instruction is added immediately after the DIVHU or DIVWU instruction in the assembly language source code. For C language source code, depending on the C compiler, the NOP instruction may automatically be added immediately after the DIVHU or DIVWU instruction is output during the build process. Confirm the specifications of the C compiler you will be using.

Remark 1. The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Remark 2. MACR: Multiply and accumulation registers (MACRH, MACRL)

Table 5 - 7 Operation List of RL78-S2 Core (13/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Increment and decrement	INC	r	1	1	—	$r \leftarrow r + 1$	x	x	
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) + 1$	x	x	
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) + 1$	x	x	
		saddr	2	2	—	$(saddr) \leftarrow (saddr) + 1$	x	x	
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) + 1$	x	x	
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) + 1$	x	x	
	DEC	r	1	1	—	$r \leftarrow r - 1$	x	x	
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) - 1$	x	x	
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) - 1$	x	x	
		saddr	2	2	—	$(saddr) \leftarrow (saddr) - 1$	x	x	
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) - 1$	x	x	
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) - 1$	x	x	
	INCW	rp	1	1	—	$rp \leftarrow rp + 1$			
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) + 1$			
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) + 1$			
		saddrp	2	2	—	$(saddrp) \leftarrow (saddrp) + 1$			
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) + 1$			
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) + 1$			
	DECW	rp	1	1	—	$rp \leftarrow rp - 1$			
		!addr16	3	2	—	$(addr16) \leftarrow (addr16) - 1$			
		ES:!addr16	4	3	—	$(ES, addr16) \leftarrow (ES, addr16) - 1$			
		saddrp	2	2	—	$(saddrp) \leftarrow (saddrp) - 1$			
		[HL+byte]	3	2	—	$(HL + byte) \leftarrow (HL + byte) - 1$			
		ES:[HL+byte]	4	3	—	$((ES:HL) + byte) \leftarrow ((ES:HL) + byte) - 1$			
Shift	SHR	A, cnt	2	1	—	$(CY \leftarrow A_0, A_{m-1} \leftarrow A_m, A_7 \leftarrow 0) \times cnt$			x
	SHRW	AX, cnt	2	1	—	$(CY \leftarrow AX_0, AX_{m-1} \leftarrow AX_m, AX_{15} \leftarrow 0) \times cnt$			x
	SHL	A, cnt	2	1	—	$(CY \leftarrow A_7, A_m \leftarrow A_{m-1}, A_0 \leftarrow 0) \times cnt$			x
		B, cnt	2	1	—	$(CY \leftarrow B_7, B_m \leftarrow B_{m-1}, B_0 \leftarrow 0) \times cnt$			x
		C, cnt	2	1	—	$(CY \leftarrow C_7, C_m \leftarrow C_{m-1}, C_0 \leftarrow 0) \times cnt$			x
	SHLW	AX, cnt	2	1	—	$(CY \leftarrow AX_{15}, AX_m \leftarrow AX_{m-1}, AX_0 \leftarrow 0) \times cnt$			x
		BC, cnt	2	1	—	$(CY \leftarrow BC_{15}, BC_m \leftarrow BC_{m-1}, BC_0 \leftarrow 0) \times cnt$			x
	SAR	A, cnt	2	1	—	$(CY \leftarrow A_0, A_{m-1} \leftarrow A_m, A_7 \leftarrow A_7) \times cnt$			x
	SARW	AX, cnt	2	1	—	$(CY \leftarrow AX_0, AX_{m-1} \leftarrow AX_m, AX_{15} \leftarrow AX_{15}) \times cnt$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark 1. The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Remark 2. cnt indicates the bit shift count.

Table 5 - 7 Operation List of RL78-S3 Core (14/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Rotate	ROR	A, 1	2	1	—	$(CY, A7 \leftarrow A0, A_{m-1} \leftarrow A_m) \times 1$			x
	ROL	A, 1	2	1	—	$(CY, A0 \leftarrow A7, A_{m+1} \leftarrow A_m) \times 1$			x
	RORC	A, 1	2	1	—	$(CY \leftarrow A0, A7 \leftarrow CY, A_{m-1} \leftarrow A_m) \times 1$			x
	ROLC	A, 1	2	1	—	$(CY \leftarrow A7, A0 \leftarrow CY, A_{m+1} \leftarrow A_m) \times 1$			x
	ROLWC	AX, 1	2	1	—	$(CY \leftarrow AX_{15}, AX0 \leftarrow CY, AX_{m+1} \leftarrow AX_m) \times 1$			x
		BC, 1	2	1	—	$(CY \leftarrow BC_{15}, BC0 \leftarrow CY, BC_{m+1} \leftarrow BC_m) \times 1$			x
Bit manipulation	MOV1	CY, A.bit	2	1	—	$CY \leftarrow A.bit$			x
		A.bit, CY	2	1	—	$A.bit \leftarrow CY$			
		CY, PSW.bit	3	1	—	$CY \leftarrow PSW.bit$			x
		PSW.bit, CY	3	4	—	$PSW.bit \leftarrow CY$	x	x	
		CY, saddr.bit	3	1	—	$CY \leftarrow (saddr).bit$			x
		saddr.bit, CY	3	2	—	$(saddr).bit \leftarrow CY$			
		CY, sfr.bit	3	1	—	$CY \leftarrow sfr.bit$			x
		sfr.bit, CY	3	2	—	$sfr.bit \leftarrow CY$			
		CY, [HL].bit	2	1	4	$CY \leftarrow (HL).bit$			x
		[HL].bit, CY	2	2	—	$(HL).bit \leftarrow CY$			
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow (ES, HL).bit$			x
		ES:[HL].bit, CY	3	3	—	$(ES, HL).bit \leftarrow CY$			
	AND1	CY, A.bit	2	1	—	$CY \leftarrow CY \wedge A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \wedge PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \wedge (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \wedge sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \wedge (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \wedge (ES, HL).bit$			x
	OR1	CY, A.bit	2	1	—	$CY \leftarrow CY \vee A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \vee PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \vee (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \vee sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \vee (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \vee (ES, HL).bit$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (15/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Bit manipulation	XOR1	CY, A.bit	2	1	—	$CY \leftarrow CY \vee A.bit$			x
		CY, PSW.bit	3	1	—	$CY \leftarrow CY \vee PSW.bit$			x
		CY, saddr.bit	3	1	—	$CY \leftarrow CY \vee (saddr).bit$			x
		CY, sfr.bit	3	1	—	$CY \leftarrow CY \vee sfr.bit$			x
		CY, [HL].bit	2	1	4	$CY \leftarrow CY \vee (HL).bit$			x
		CY, ES:[HL].bit	3	2	5	$CY \leftarrow CY \vee (ES, HL).bit$			x
	SET1	A.bit	2	1	—	$A.bit \leftarrow 1$			
		PSW.bit	3	4	—	$PSW.bit \leftarrow 1$	x	x	x
		!addr16.bit	4	2	—	$(addr16).bit \leftarrow 1$			
		ES:!addr16.bit	5	3	—	$(ES, addr16).bit \leftarrow 1$			
		saddr.bit	3	2	—	$(saddr).bit \leftarrow 1$			
		sfr.bit	3	2	—	$sfr.bit \leftarrow 1$			
		[HL].bit	2	2	—	$(HL).bit \leftarrow 1$			
		ES:[HL].bit	3	3	—	$(ES, HL).bit \leftarrow 1$			
	CLR1	A.bit	2	1	—	$A.bit \leftarrow 0$			
		PSW.bit	3	4	—	$PSW.bit \leftarrow 0$	x	x	x
		!addr16.bit	4	2	—	$(addr16).bit \leftarrow 0$			
		ES:!addr16.bit	5	3	—	$(ES, addr16).bit \leftarrow 0$			
		saddr.bit	3	2	—	$(saddr).bit \leftarrow 0$			
		sfr.bit	3	2	—	$sfr.bit \leftarrow 0$			
		[HL].bit	2	2	—	$(HL).bit \leftarrow 0$			
		ES:[HL].bit	3	3	—	$(ES, HL).bit \leftarrow 0$			
	SET1	CY	2	1	—	$CY \leftarrow 1$			1
	CLR1	CY	2	1	—	$CY \leftarrow 0$			0
	NOT1	CY	2	1	—	$CY \leftarrow \overline{CY}$			x

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (16/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Call and return	CALL	rp	2	3	—	$(SP - 2) \leftarrow (PC + 2)_S, (SP - 3) \leftarrow (PC + 2)_H,$ $(SP - 4) \leftarrow (PC + 2)_L, PC \leftarrow CS, rp,$ $SP \leftarrow SP - 4$			
		\$!addr20	3	3	—	$(SP - 2) \leftarrow (PC + 3)_S, (SP - 3) \leftarrow (PC + 3)_H,$ $(SP - 4) \leftarrow (PC + 3)_L, PC \leftarrow PC + 3 + jdisp16,$ $SP \leftarrow SP - 4$			
		!addr16	3	3	—	$(SP - 2) \leftarrow (PC + 3)_S, (SP - 3) \leftarrow (PC + 3)_H,$ $(SP - 4) \leftarrow (PC + 3)_L, PC \leftarrow 0000, addr16,$ $SP \leftarrow SP - 4$			
		!!addr20	4	3	—	$(SP - 2) \leftarrow (PC + 4)_S, (SP - 3) \leftarrow (PC + 4)_H,$ $(SP - 4) \leftarrow (PC + 4)_L, PC \leftarrow addr20,$ $SP \leftarrow SP - 4$			
	CALLT	[addr5]	2	5	—	$(SP - 2) \leftarrow (PC + 2)_S, (SP - 3) \leftarrow (PC + 2)_H,$ $(SP - 4) \leftarrow (PC + 2)_L, PCs \leftarrow 0000,$ $PCH \leftarrow (0000, addr5 + 1),$ $PCL \leftarrow (0000, addr5),$ $SP \leftarrow SP - 4$			
	BRK	—	2	5	—	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow (PC + 2)_S,$ $(SP - 3) \leftarrow (PC + 2)_H, (SP - 4) \leftarrow (PC + 2)_L,$ $PCs \leftarrow 0000,$ $PCH \leftarrow (0007FH), PCL \leftarrow (0007EH),$ $SP \leftarrow SP - 4, IE \leftarrow 0$			
	RET	—	1	6	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), SP \leftarrow SP + 4$			
	RETI	—	2	6	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), PSW \leftarrow (SP + 3),$ $SP \leftarrow SP + 4$	R	R	R
	RETB	—	2	6	—	$PCL \leftarrow (SP), PCH \leftarrow (SP + 1),$ $PCs \leftarrow (SP + 2), PSW \leftarrow (SP + 3),$ $SP \leftarrow SP + 4$	R	R	R

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (17/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Stack manipulation	PUSH	PSW	2	1	—	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow 00H, SP \leftarrow SP - 2$			
		rp	1	1	—	$(SP - 1) \leftarrow rpH, (SP - 2) \leftarrow rpL, SP \leftarrow SP - 2$			
	POP	PSW	2	3	—	$PSW \leftarrow (SP + 1), SP \leftarrow SP + 2$	R	R	R
		rp	1	1	—	$rpL \leftarrow (SP), rpH \leftarrow (SP + 1), SP \leftarrow SP + 2$			
	MOVW	SP, #word	4	1	—	$SP \leftarrow word$			
		SP, AX	2	1	—	$SP \leftarrow AX$			
		AX, SP	2	1	—	$AX \leftarrow SP$			
		HL, SP	3	1	—	$HL \leftarrow SP$			
		BC, SP	3	1	—	$BC \leftarrow SP$			
		DE, SP	3	1	—	$DE \leftarrow SP$			
	ADDW	SP, #byte	2	1	—	$SP \leftarrow SP + byte$			
	SUBW	SP, #byte	2	1	—	$SP \leftarrow SP - byte$			
Unconditional branch	BR	AX	2	3	—	$PC \leftarrow CS, AX$			
		\$addr20	2	3	—	$PC \leftarrow PC + 2 + jdisp8$			
		\$!addr20	3	3	—	$PC \leftarrow PC + 3 + jdisp16$			
		!addr16	3	3	—	$PC \leftarrow 0000, addr16$			
		!addr20	4	3	—	$PC \leftarrow addr20$			
Conditional branch	BC	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 1$			
	BNC	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 0$			
	BZ	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 1$			
	BNZ	\$addr20	2	2/4 Note 3	—	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 0$			
	BH	\$addr20	3	2/4 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if $(Z \vee CY) = 0$			
	BNH	\$addr20	3	2/4 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if $(Z \vee CY) = 1$			
	BT	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 1			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 1			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 1			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 1			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. These indicate the numbers of clock cycles when the condition is not and is met.

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

Table 5 - 7 Operation List of RL78-S3 Core (18/18)

Instruction Group	Mnemonic	Operands	Bytes	Clock Cycles		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Conditional branch	BT	[HL].bit, \$addr20	3	3/5 Note 3	6/7	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 1			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	7/8	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 1			
	BF	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 0			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 0			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 0			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 0			
		[HL].bit, \$addr20	3	3/5 Note 3	6/7	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 0			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	7/8	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 0			
	BTCLR	saddr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (saddr).bit = 1 then reset (saddr).bit			
		sfr.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if sfr.bit = 1 then reset sfr.bit			
		A.bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if A.bit = 1 then reset A.bit			
		PSW.bit, \$addr20	4	3/5 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if PSW.bit = 1 then reset PSW.bit	x	x	x
		[HL].bit, \$addr20	3	3/5 Note 3	—	$PC \leftarrow PC + 3 + jdisp8$ if (HL).bit = 1 then reset (HL).bit			
		ES:[HL].bit, \$addr20	4	4/6 Note 3	—	$PC \leftarrow PC + 4 + jdisp8$ if (ES, HL).bit = 1 then reset (ES, HL).bit			
Conditional skip	SKC	—	2	1	—	Next instruction skip if CY = 1			
	SKNC	—	2	1	—	Next instruction skip if CY = 0			
	SKZ	—	2	1	—	Next instruction skip if Z = 1			
	SKNZ	—	2	1	—	Next instruction skip if Z = 0			
	SKH	—	2	1	—	Next instruction skip if $(Z \vee CY) = 0$			
	SKNH	—	2	1	—	Next instruction skip if $(Z \vee CY) = 1$			
CPU control	SEL ^{Note 4}	RBn	2	1	—	$RBS[1:0] \leftarrow n$			
	NOP	—	1	1	—	No Operation			
	EI	—	3	4	—	$IE \leftarrow 1$ (Enable Interrupt)			
	DI	—	3	4	—	$IE \leftarrow 0$ (Disable Interrupt)			
	HALT	—	2	3	—	Set HALT Mode			
	STOP	—	2	3	—	Set STOP Mode			

Note 1. These are the numbers of CPU clock cycles (fCLK) in the case of access to the internal RAM area, SFR area, or 2nd SFR area, or when the instruction does not involve data access.

Note 2. These are the numbers of CPU clock cycles (fCLK) in the case of access to the code flash memory area, or of access to the data flash memory area by an 8-bit instruction.

Note 3. These indicate the numbers of clock cycles when the condition is not and is met.

Note 4. n indicates the number of register banks (n = 0 to 3).

Remark The listed numbers of clock cycles are for when the program is present in the internal ROM (flash memory) area. If the instruction is fetched from the internal RAM area, the number becomes up to double the listed number plus 3 clock cycles.

5.6 Instruction Format

Instructions consist of fixed opcodes followed by operands. Their formats are listed below.

Table 5 - 8 List of Instruction Formats (1/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOV	X, #byte	50	data	—	—	—
	A, #byte	51	data	—	—	—
	C, #byte	52	data	—	—	—
	B, #byte	53	data	—	—	—
	E, #byte	54	data	—	—	—
	D, #byte	55	data	—	—	—
	L, #byte	56	data	—	—	—
	H, #byte	57	data	—	—	—
	saddr, #byte	CD	saddr	data	—	—
	sfr, #byte	CE	sfr	data	—	—
	!addr16, #byte	CF	adrl	adrh	data	—
	A, X	60	—	—	—	—
	A, C	62	—	—	—	—
	A, B	63	—	—	—	—
	A, E	64	—	—	—	—
	A, D	65	—	—	—	—
	A, L	66	—	—	—	—
	A, H	67	—	—	—	—
	X, A	70	—	—	—	—
	C, A	72	—	—	—	—
	B, A	73	—	—	—	—
	E, A	74	—	—	—	—
	D, A	75	—	—	—	—
	L, A	76	—	—	—	—
	H, A	77	—	—	—	—
	A, saddr	8D	saddr	—	—	—
	saddr, A	9D	saddr	—	—	—
	A, sfr	8E	sfr	—	—	—
	sfr, A	9E	sfr	—	—	—
	A, !addr16	8F	adrl	adrh	—	—
	!addr16, A	9F	adrl	adrh	—	—
	PSW, #byte	CE	FA	data	—	—
	A, PSW	8E	FA	—	—	—
	PSW, A	9E	FA	—	—	—
	ES, #byte	41	data	—	—	—
	ES, saddr	61	B8	saddr	—	—

Table 5 - 8 List of Instruction Formats (2/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOV	A, ES	8E	FD	—	—	—
	ES, A	9E	FD	—	—	—
	CS, #byte	CE	FC	data	—	—
	A, CS	8E	FC	—	—	—
	CS, A	9E	FC	—	—	—
	A, [DE]	89	—	—	—	—
	[DE], A	99	—	—	—	—
	[DE+byte], #byte	CA	adr	data	—	—
	A, [DE+byte]	8A	adr	—	—	—
	[DE+byte], A	9A	adr	—	—	—
	A, [HL]	8B	—	—	—	—
	[HL], A	9B	—	—	—	—
	[HL+byte], #byte	CC	adr	data	—	—
	A, [HL+byte]	8C	adr	—	—	—
	[HL+byte], A	9C	adr	—	—	—
	A, [HL+B]	61	C9	—	—	—
	[HL+B], A	61	D9	—	—	—
	A, [HL+C]	61	E9	—	—	—
	[HL+C], A	61	F9	—	—	—
	word[B], #byte	19	adrl	adrh	data	—
	A, word[B]	09	adrl	adrh	—	—
	word[B], A	18	adrl	adrh	—	—
	word[C], #byte	38	adrl	adrh	data	—
	A, word[C]	29	adrl	adrh	—	—
	word[C], A	28	adrl	adrh	—	—
	word[BC], #byte	39	adrl	adrh	data	—
	A, word[BC]	49	adrl	adrh	—	—
	word[BC], A	48	adrl	adrh	—	—
	[SP+byte], #byte	C8	adr	data	—	—
	A, [SP+byte]	88	adr	—	—	—
	[SP+byte], A	98	adr	—	—	—
	B, saddr	E8	saddr	—	—	—
	B, !addr16	E9	adrl	adrh	—	—
	C, saddr	F8	saddr	—	—	—
	C, !addr16	F9	adrl	adrh	—	—
	X, saddr	D8	saddr	—	—	—
	X, !addr16	D9	adrl	adrh	—	—
	ES:!addr16, #byte	11	CF	adrl	adrh	data
	A, ES:!addr16	11	8F	adrl	adrh	—

Table 5 - 8 List of Instruction Formats (3/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOV	ES:!addr16, A	11	9F	adrl	adrh	—
	A, ES:[DE]	11	89	—	—	—
	ES:[DE], A	11	99	—	—	—
	ES:[DE+byte], #byte	11	CA	adr	data	—
	A, ES:[DE+byte]	11	8A	adr	—	—
	ES:[DE+byte], A	11	9A	adr	—	—
	A, ES:[HL]	11	8B	—	—	—
	ES:[HL], A	11	9B	—	—	—
	ES:[HL+byte], #byte	11	CC	adr	data	—
	A, ES:[HL+byte]	11	8C	adr	—	—
	ES:[HL+byte], A	11	9C	adr	—	—
	A, ES:[HL+B]	11	61	C9	—	—
	ES:[HL+B], A	11	61	D9	—	—
	A, ES:[HL+C]	11	61	E9	—	—
	ES:[HL+C], A	11	61	F9	—	—
	ES:word[B], #byte	11	19	adrl	adrh	data
	A, ES:word[B]	11	09	adrl	adrh	—
	ES:word[B], A	11	18	adrl	adrh	—
	ES:word[C], #byte	11	38	adrl	adrh	data
	A, ES:word[C]	11	29	adrl	adrh	—
	ES:word[C], A	11	28	adrl	adrh	—
	ES:word[BC], #byte	11	39	adrl	adrh	data
	A, ES:word[BC]	11	49	adrl	adrh	—
	ES:word[BC], A	11	48	adrl	adrh	—
	B, ES:!addr16	11	E9	adrl	adrh	—
	C, ES:!addr16	11	F9	adrl	adrh	—
	X, ES:!addr16	11	D9	adrl	adrh	—
XCH	A, X	08	—	—	—	—
	A, C	61	8A	—	—	—
	A, B	61	8B	—	—	—
	A, E	61	8C	—	—	—
	A, D	61	8D	—	—	—
	A, L	61	8E	—	—	—
	A, H	61	8F	—	—	—
	A, saddr	61	A8	saddr	—	—
	A, sfr	61	AB	sfr	—	—
	A, !addr16	61	AA	adrl	adrh	—
	A, [DE]	61	AE	—	—	—
	A, [DE+byte]	61	AF	adr	—	—

Table 5 - 8 List of Instruction Formats (4/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
XCH	A, [HL]	61	AC	—	—	—
	A, [HL+byte]	61	AD	adr	—	—
	A, [HL+B]	61	B9	—	—	—
	A, [HL+C]	61	A9	—	—	—
	A, ES:!addr16	11	61	AA	adrl	adrh
	A, ES:[DE]	11	61	AE	—	—
	A, ES:[DE+byte]	11	61	AF	adr	—
	A, ES:[HL]	11	61	AC	—	—
	A, ES:[HL+byte]	11	61	AD	adr	—
	A, ES:[HL+B]	11	61	B9	—	—
	A, ES:[HL+C]	11	61	A9	—	—
ONEB	A	E1	—	—	—	—
	X	E0	—	—	—	—
	B	E3	—	—	—	—
	C	E2	—	—	—	—
	saddr	E4	saddr	—	—	—
	!addr16	E5	adrl	adrh	—	—
	ES:!addr16	11	E5	adrl	adrh	—
CLRB	A	F1	—	—	—	—
	X	F0	—	—	—	—
	B	F3	—	—	—	—
	C	F2	—	—	—	—
	saddr	F4	saddr	—	—	—
	!addr16	F5	adrl	adrh	—	—
	ES:!addr16	11	F5	adrl	adrh	—
MOVS	[HL+byte], X	61	CE	adr	—	—
	ES:[HL+byte], X	11	61	CE	adr	—
MOVW	AX, #word	30	datal	datah	—	—
	BC, #word	32	datal	datah	—	—
	DE, #word	34	datal	datah	—	—
	HL, #word	36	datal	datah	—	—
	saddrp, #word	C9	saddr	datal	datah	—
	sfrp, #word	CB	sfr	datal	datah	—
	AX, saddrp	AD	saddr	—	—	—
	saddrp, AX	BD	saddr	—	—	—
	AX, sfrp	AE	sfr	—	—	—
	sfrp, AX	BE	sfr	—	—	—
	AX, BC	13	—	—	—	—
	AX, DE	15	—	—	—	—

Table 5 - 8 List of Instruction Formats (5/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOVW	AX, HL	17	—	—	—	—
	BC, AX	12	—	—	—	—
	DE, AX	14	—	—	—	—
	HL, AX	16	—	—	—	—
	AX, !addr16	AF	adrl	adrh	—	—
	!addr16, AX	BF	adrl	adrh	—	—
	AX, [DE]	A9	—	—	—	—
	[DE], AX	B9	—	—	—	—
	AX, [DE+byte]	AA	adr	—	—	—
	[DE+byte], AX	BA	adr	—	—	—
	AX, [HL]	AB	—	—	—	—
	[HL], AX	BB	—	—	—	—
	AX, [HL+byte]	AC	adr	—	—	—
	[HL+byte], AX	BC	adr	—	—	—
	AX, word[B]	59	adrl	adrh	—	—
	word[B], AX	58	adrl	adrh	—	—
	AX, word[C]	69	adrl	adrh	—	—
	word[C], AX	68	adrl	adrh	—	—
	AX, word[BC]	79	adrl	adrh	—	—
	word[BC], AX	78	adrl	adrh	—	—
	AX, [SP+byte]	A8	adr	—	—	—
	[SP+byte], AX	B8	adr	—	—	—
	BC, saddrp	DA	saddr	—	—	—
	BC, !addr16	DB	adrl	adrh	—	—
	DE, saddrp	EA	saddr	—	—	—
	DE, !addr16	EB	adrl	adrh	—	—
	HL, saddrp	FA	saddr	—	—	—
	HL, !addr16	FB	adrl	adrh	—	—
	AX, ES:!addr16	11	AF	adrl	adrh	—
	ES:!addr16, AX	11	BF	adrl	adrh	—
	AX, ES:[DE]	11	A9	—	—	—
	ES:[DE], AX	11	B9	—	—	—
	AX, ES:[DE+byte]	11	AA	adr	—	—
	ES:[DE+byte], AX	11	BA	adr	—	—
	AX, ES:[HL]	11	AB	—	—	—
	ES:[HL], AX	11	BB	—	—	—
	AX, ES:[HL+byte]	11	AC	adr	—	—
	ES:[HL+byte], AX	11	BC	adr	—	—
	AX, ES:word[B]	11	59	adrl	adrh	—

Table 5 - 8 List of Instruction Formats (6/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOVW	ES:word[B], AX	11	58	adrl	adrh	—
	AX, ES:word[C]	11	69	adrl	adrh	—
	ES:word[C], AX	11	68	adrl	adrh	—
	AX, ES:word[BC]	11	79	adrl	adrh	—
	ES:word[BC], AX	11	78	adrl	adrh	—
	BC, ES:!addr16	11	DB	adrl	adrh	—
	DE, ES:!addr16	11	EB	adrl	adrh	—
	HL, ES:!addr16	11	FB	adrl	adrh	—
XCHW	AX, BC	33	—	—	—	—
	AX, DE	35	—	—	—	—
	AX, HL	37	—	—	—	—
ONEW	AX	E6	—	—	—	—
	BC	E7	—	—	—	—
CLRW	AX	F6	—	—	—	—
	BC	F7	—	—	—	—
ADD	A, #byte	0C	data	—	—	—
	saddr, #byte	0A	saddr	data	—	—
	A, X	61	08	—	—	—
	A, C	61	0A	—	—	—
	A, B	61	0B	—	—	—
	A, E	61	0C	—	—	—
	A, D	61	0D	—	—	—
	A, L	61	0E	—	—	—
	A, H	61	0F	—	—	—
	X, A	61	00	—	—	—
	A, A	61	01	—	—	—
	C, A	61	02	—	—	—
	B, A	61	03	—	—	—
	E, A	61	04	—	—	—
	D, A	61	05	—	—	—
	L, A	61	06	—	—	—
	H, A	61	07	—	—	—
	A, saddr	0B	saddr	—	—	—
	A, !addr16	0F	adrl	adrh	—	—
	A, [HL]	0D	—	—	—	—
	A, [HL+byte]	0E	adr	—	—	—
	A, [HL+B]	61	80	—	—	—
	A, [HL+C]	61	82	—	—	—
	A, ES:!addr16	11	0F	adrl	adrh	—

Table 5 - 8 List of Instruction Formats (7/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
ADD	A, ES:[HL]	11	0D	—	—	—
	A, ES:[HL+byte]	11	0E	adr	—	—
	A, ES:[HL+B]	11	61	80	—	—
	A, ES:[HL+C]	11	61	82	—	—
ADDC	A, #byte	1C	data	—	—	—
	saddr, #byte	1A	saddr	data	—	—
	A, X	61	18	—	—	—
	A, C	61	1A	—	—	—
	A, B	61	1B	—	—	—
	A, E	61	1C	—	—	—
	A, D	61	1D	—	—	—
	A, L	61	1E	—	—	—
	A, H	61	1F	—	—	—
	X, A	61	10	—	—	—
	A, A	61	11	—	—	—
	C, A	61	12	—	—	—
	B, A	61	13	—	—	—
	E, A	61	14	—	—	—
	D, A	61	15	—	—	—
	L, A	61	16	—	—	—
	H, A	61	17	—	—	—
	A, saddr	1B	saddr	—	—	—
	A, !addr16	1F	adrl	adrh	—	—
	A, [HL]	1D	—	—	—	—
	A, [HL+byte]	1E	adr	—	—	—
	A, [HL+B]	61	90	—	—	—
	A, [HL+C]	61	92	—	—	—
	A, ES:!addr16	11	1F	adrl	adrh	—
	A, ES:[HL]	11	1D	—	—	—
	A, ES:[HL+byte]	11	1E	adr	—	—
	A, ES:[HL+B]	11	61	90	—	—
	A, ES:[HL+C]	11	61	92	—	—
SUB	A, #byte	2C	data	—	—	—
	saddr, #byte	2A	saddr	data	—	—
	A, X	61	28	—	—	—
	A, C	61	2A	—	—	—
	A, B	61	2B	—	—	—
	A, E	61	2C	—	—	—
	A, D	61	2D	—	—	—

Table 5 - 8 List of Instruction Formats (8/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
SUB	A, L	61	2E	—	—	—
	A, H	61	2F	—	—	—
	X, A	61	20	—	—	—
	A, A	61	21	—	—	—
	C, A	61	22	—	—	—
	B, A	61	23	—	—	—
	E, A	61	24	—	—	—
	D, A	61	25	—	—	—
	L, A	61	26	—	—	—
	H, A	61	27	—	—	—
	A, saddr	2B	saddr	—	—	—
	A, !addr16	2F	adrl	adrh	—	—
	A, [HL]	2D	—	—	—	—
	A, [HL+byte]	2E	adr	—	—	—
	A, [HL+B]	61	A0	—	—	—
	A, [HL+C]	61	A2	—	—	—
	A, ES:!addr16	11	2F	adrl	adrh	—
	A, ES:[HL]	11	2D	—	—	—
	A, ES:[HL+byte]	11	2E	adr	—	—
	A, ES:[HL+B]	11	61	A0	—	—
	A, ES:[HL+C]	11	61	A2	—	—
SUBC	A, #byte	3C	data	—	—	—
	saddr, #byte	3A	saddr	data	—	—
	A, X	61	38	—	—	—
	A, C	61	3A	—	—	—
	A, B	61	3B	—	—	—
	A, E	61	3C	—	—	—
	A, D	61	3D	—	—	—
	A, L	61	3E	—	—	—
	A, H	61	3F	—	—	—
	X, A	61	30	—	—	—
	A, A	61	31	—	—	—
	C, A	61	32	—	—	—
	B, A	61	33	—	—	—
	E, A	61	34	—	—	—
	D, A	61	35	—	—	—
	L, A	61	36	—	—	—
	H, A	61	37	—	—	—
	A, saddr	3B	saddr	—	—	—

Table 5 - 8 List of Instruction Formats (9/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
SUBC	A, !addr16	3F	adrl	adrh	—	—
	A, [HL]	3D	—	—	—	—
	A, [HL+byte]	3E	adr	—	—	—
	A, [HL+B]	61	B0	—	—	—
	A, [HL+C]	61	B2	—	—	—
	A, ES:!addr16	11	3F	adrl	adrh	—
	A, ES:[HL]	11	3D	—	—	—
	A, ES:[HL+byte]	11	3E	adr	—	—
	A, ES:[HL+B]	11	61	B0	—	—
	A, ES:[HL+C]	11	61	B2	—	—
AND	A, #byte	5C	data	—	—	—
	saddr, #byte	5A	saddr	data	—	—
	A, X	61	58	—	—	—
	A, C	61	5A	—	—	—
	A, B	61	5B	—	—	—
	A, E	61	5C	—	—	—
	A, D	61	5D	—	—	—
	A, L	61	5E	—	—	—
	A, H	61	5F	—	—	—
	X, A	61	50	—	—	—
	A, A	61	51	—	—	—
	C, A	61	52	—	—	—
	B, A	61	53	—	—	—
	E, A	61	54	—	—	—
	D, A	61	55	—	—	—
	L, A	61	56	—	—	—
	H, A	61	57	—	—	—
	A, saddr	5B	saddr	—	—	—
	A, !addr16	5F	adrl	adrh	—	—
	A, [HL]	5D	—	—	—	—
	A, [HL+byte]	5E	adr	—	—	—
	A, [HL+B]	61	D0	—	—	—
	A, [HL+C]	61	D2	—	—	—
	A, ES:!addr16	11	5F	adrl	adrh	—
	A, ES:[HL]	11	5D	—	—	—
	A, ES:[HL+byte]	11	5E	adr	—	—
	A, ES:[HL+B]	11	61	D0	—	—
	A, ES:[HL+C]	11	61	D2	—	—

Table 5 - 8 List of Instruction Formats (10/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
OR	A, #byte	6C	data	—	—	—
	saddr, #byte	6A	saddr	data	—	—
	A, X	61	68	—	—	—
	A, C	61	6A	—	—	—
	A, B	61	6B	—	—	—
	A, E	61	6C	—	—	—
	A, D	61	6D	—	—	—
	A, L	61	6E	—	—	—
	A, H	61	6F	—	—	—
	X, A	61	60	—	—	—
	A, A	61	61	—	—	—
	C, A	61	62	—	—	—
	B, A	61	63	—	—	—
	E, A	61	64	—	—	—
	D, A	61	65	—	—	—
	L, A	61	66	—	—	—
	H, A	61	67	—	—	—
	A, saddr	6B	saddr	—	—	—
	A, !addr16	6F	adrl	adrh	—	—
	A, [HL]	6D	—	—	—	—
	A, [HL+byte]	6E	adr	—	—	—
	A, [HL+B]	61	E0	—	—	—
	A, [HL+C]	61	E2	—	—	—
	A, ES:!addr16	11	6F	adrl	adrh	—
	A, ES:[HL]	11	6D	—	—	—
	A, ES:[HL+byte]	11	6E	adr	—	—
	A, ES:[HL+B]	11	61	E0	—	—
	A, ES:[HL+C]	11	61	E2	—	—
XOR	A, #byte	7C	data	—	—	—
	saddr, #byte	7A	saddr	data	—	—
	A, X	61	78	—	—	—
	A, C	61	7A	—	—	—
	A, B	61	7B	—	—	—
	A, E	61	7C	—	—	—
	A, D	61	7D	—	—	—
	A, L	61	7E	—	—	—
	A, H	61	7F	—	—	—
	X, A	61	70	—	—	—
	A, A	61	71	—	—	—

Table 5 - 8 List of Instruction Formats (11/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
XOR	C, A	61	72	—	—	—
	B, A	61	73	—	—	—
	E, A	61	74	—	—	—
	D, A	61	75	—	—	—
	L, A	61	76	—	—	—
	H, A	61	77	—	—	—
	A, saddr	7B	saddr	—	—	—
	A, !addr16	7F	adrl	adrh	—	—
	A, [HL]	7D		—	—	—
	A, [HL+byte]	7E	adr	—	—	—
	A, [HL+B]	61	F0	—	—	—
	A, [HL+C]	61	F2	—	—	—
	A, ES:!addr16	11	7F	adrl	adrh	—
	A, ES:[HL]	11	7D	—	—	—
	A, ES:[HL+byte]	11	7E	adr	—	—
	A, ES:[HL+B]	11	61	F0	—	—
	A, ES:[HL+C]	11	61	F2	—	—
CMP	A, #byte	4C	data	—	—	—
	saddr, #byte	4A	saddr	data	—	—
	A, X	61	48	—	—	—
	A, C	61	4A	—	—	—
	A, B	61	4B	—	—	—
	A, E	61	4C	—	—	—
	A, D	61	4D	—	—	—
	A, L	61	4E	—	—	—
	A, H	61	4F	—	—	—
	X, A	61	40	—	—	—
	A, A	61	41	—	—	—
	C, A	61	42	—	—	—
	B, A	61	43	—	—	—
	E, A	61	44	—	—	—
	D, A	61	45	—	—	—
	L, A	61	46	—	—	—
	H, A	61	47	—	—	—
	A, saddr	4B	saddr	—	—	—
	A, !addr16	4F	adrl	adrh	—	—
	A, [HL]	4D		—	—	—
	A, [HL+byte]	4E	adr	—	—	—
	A, [HL+B]	61	C0	—	—	—

Table 5 - 8 List of Instruction Formats (12/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
CMP	A, [HL+C]	61	C2	—	—	—
	!addr16, #byte	40	adrl	adrh	data	—
	A, ES:!addr16	11	4F	adrl	adrh	—
	A, ES:[HL]	11	4D	—	—	—
	A, ES:[HL+byte]	11	4E	adr	—	—
	A, ES:[HL+B]	11	61	C0	—	—
	A, ES:[HL+C]	11	61	C2	—	—
	ES:!addr16, #byte	11	40	adrl	adrh	data
CMP0	A	D1	—	—	—	—
	X	D0	—	—	—	—
	B	D3	—	—	—	—
	C	D2	—	—	—	—
	saddr	D4	saddr	—	—	—
	!addr16	D5	adrl	adrh	—	—
	ES:!addr16	11	D5	adrl	adrh	—
CMPS	X, [HL+byte]	61	DE	adr	—	—
	X, ES:[HL+byte]	11	61	DE	adr	—
ADDW	AX, #word	04	data1	datah	—	—
	AX, AX	01	—	—	—	—
	AX, BC	03	—	—	—	—
	AX, DE	05	—	—	—	—
	AX, HL	07	—	—	—	—
	AX, saddrp	06	saddr	—	—	—
	AX, !addr16	02	adrl	adrh	—	—
	AX, [HL+byte]	61	09	adr	—	—
	AX, ES:!addr16	11	02	adrl	adrh	—
	AX, ES:[HL+byte]	11	61	09	adr	—
SUBW	AX, #word	24	data1	datah	—	—
	AX, BC	23	—	—	—	—
	AX, DE	25	—	—	—	—
	AX, HL	27	—	—	—	—
	AX, saddrp	26	saddr	—	—	—
	AX, !addr16	22	adrl	adrh	—	—
	AX, [HL+byte]	61	29	adr	—	—
	AX, ES:!addr16	11	22	adrl	adrh	—
	AX, ES:[HL+byte]	11	61	29	adr	—
CMPW	AX, #word	44	data1	datah	—	—
	AX, BC	43	—	—	—	—
	AX, DE	45	—	—	—	—

Table 5 - 8 List of Instruction Formats (13/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
CMPW	AX, HL	47	—	—	—	—
	AX, saddrp	46	saddr	—	—	—
	AX, !addr16	42	adrl	adrh	—	—
	AX, [HL+byte]	61	49	adr	—	—
	AX, ES:!addr16	11	42	adrl	adrh	—
	AX, ES:[HL+byte]	11	61	49	adr	—
MULU	X	D6	—	—	—	—
MULHU ^{Note 1}		CE	FB	01	—	—
MULH ^{Note 1}		CE	FB	02	—	—
DIVHU ^{Note 1}		CE	FB	03	—	—
DIVWU ^{Note 1}		CE	FB	0B	—	—
MACHU ^{Note 1}		CE	FB	05	—	—
MACH ^{Note 1}		CE	FB	06	—	—
INC	X	80	—	—	—	—
	A	81	—	—	—	—
	C	82	—	—	—	—
	B	83	—	—	—	—
	E	84	—	—	—	—
	D	85	—	—	—	—
	L	86	—	—	—	—
	H	87	—	—	—	—
	saddr	A4	saddr	—	—	—
	!addr16	A0	adrl	adrh	—	—
	[HL+byte]	61	59	adr	—	—
	ES:!addr16	11	A0	adrl	adrh	—
	ES:[HL+byte]	11	61	59	adr	—
DEC	X	90	—	—	—	—
	A	91	—	—	—	—
	C	92	—	—	—	—
	B	93	—	—	—	—
	E	94	—	—	—	—
	D	95	—	—	—	—
	L	96	—	—	—	—
	H	97	—	—	—	—
	saddr	B4	saddr	—	—	—
	!addr16	B0	adrl	adrh	—	—
	[HL+byte]	61	69	adr	—	—
	ES:!addr16	11	B0	adrl	adrh	—
	ES:[HL+byte]	11	61	69	adr	—

Table 5 - 8 List of Instruction Formats (14/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
INCW	AX	A1	—	—	—	—
	BC	A3	—	—	—	—
	DE	A5	—	—	—	—
	HL	A7	—	—	—	—
	saddrp	A6	saddr	—	—	—
	!addr16	A2	adrl	adrh	—	—
	[HL+byte]	61	79	adr	—	—
	ES:!addr16	11	A2	adrl	adrh	—
	ES:[HL+byte]	11	61	79	adr	—
DECW	AX	B1	—	—	—	—
	BC	B3	—	—	—	—
	DE	B5	—	—	—	—
	HL	B7	—	—	—	—
	saddrp	B6	saddr	—	—	—
	!addr16	B2	adrl	adrh	—	—
	[HL+byte]	61	89	adr	—	—
	ES:!addr16	11	B2	adrl	adrh	—
	ES:[HL+byte]	11	61	89	adr	—
SHR	A, 1	31	1A	—	—	—
	A, 2	31	2A	—	—	—
	A, 3	31	3A	—	—	—
	A, 4	31	4A	—	—	—
	A, 5	31	5A	—	—	—
	A, 6	31	6A	—	—	—
	A, 7	31	7A	—	—	—
SHRW	AX, 1	31	1E	—	—	—
	AX, 2	31	2E	—	—	—
	AX, 3	31	3E	—	—	—
	AX, 4	31	4E	—	—	—
	AX, 5	31	5E	—	—	—
	AX, 6	31	6E	—	—	—
	AX, 7	31	7E	—	—	—
	AX, 8	31	8E	—	—	—
	AX, 9	31	9E	—	—	—
	AX, 10	31	AE	—	—	—
	AX, 11	31	BE	—	—	—
	AX, 12	31	CE	—	—	—

Table 5 - 8 List of Instruction Formats (15/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
SHRW	AX, 13	31	DE	—	—	—
	AX, 14	31	EE	—	—	—
	AX, 15	31	FE	—	—	—
SHL	A, 1	31	19	—	—	—
	A, 2	31	29	—	—	—
	A, 3	31	39	—	—	—
	A, 4	31	49	—	—	—
	A, 5	31	59	—	—	—
	A, 6	31	69	—	—	—
	A, 7	31	79	—	—	—
	B, 1	31	18	—	—	—
	B, 2	31	28	—	—	—
	B, 3	31	38	—	—	—
	B, 4	31	48	—	—	—
	B, 5	31	58	—	—	—
	B, 6	31	68	—	—	—
	B, 7	31	78	—	—	—
	C, 1	31	17	—	—	—
	C, 2	31	27	—	—	—
	C, 3	31	37	—	—	—
	C, 4	31	47	—	—	—
	C, 5	31	57	—	—	—
	C, 6	31	67	—	—	—
	C, 7	31	77	—	—	—
SHLW	AX, 1	31	1D	—	—	—
	AX, 2	31	2D	—	—	—
	AX, 3	31	3D	—	—	—
	AX, 4	31	4D	—	—	—
	AX, 5	31	5D	—	—	—
	AX, 6	31	6D	—	—	—
	AX, 7	31	7D	—	—	—
	AX, 8	31	8D	—	—	—
	AX, 9	31	9D	—	—	—
	AX, 10	31	AD	—	—	—
	AX, 11	31	BD	—	—	—
	AX, 12	31	CD	—	—	—
	AX, 13	31	DD	—	—	—
	AX, 14	31	ED	—	—	—
	AX, 15	31	FD	—	—	—

Table 5 - 8 List of Instruction Formats (16/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
SHLW	BC, 1	31	1C	—	—	—
	BC, 2	31	2C	—	—	—
	BC, 3	31	3C	—	—	—
	BC, 4	31	4C	—	—	—
	BC, 5	31	5C	—	—	—
	BC, 6	31	6C	—	—	—
	BC, 7	31	7C	—	—	—
	BC, 8	31	8C	—	—	—
	BC, 9	31	9C	—	—	—
	BC, 10	31	AC	—	—	—
	BC, 11	31	BC	—	—	—
	BC, 12	31	CC	—	—	—
	BC, 13	31	DC	—	—	—
	BC, 14	31	EC	—	—	—
	BC, 15	31	FC	—	—	—
SAR	A, 1	31	1B	—	—	—
	A, 2	31	2B	—	—	—
	A, 3	31	3B	—	—	—
	A, 4	31	4B	—	—	—
	A, 5	31	5B	—	—	—
	A, 6	31	6B	—	—	—
	A, 7	31	7B	—	—	—
SARW	AX, 1	31	1F	—	—	—
	AX, 2	31	2F	—	—	—
	AX, 3	31	3F	—	—	—
	AX, 4	31	4F	—	—	—
	AX, 5	31	5F	—	—	—
	AX, 6	31	6F	—	—	—
	AX, 7	31	7F	—	—	—
	AX, 8	31	8F	—	—	—
	AX, 9	31	9F	—	—	—
	AX, 10	31	AF	—	—	—
	AX, 11	31	BF	—	—	—
	AX, 12	31	CF	—	—	—
	AX, 13	31	DF	—	—	—
	AX, 14	31	EF	—	—	—
	AX, 15	31	FF	—	—	—
ROR	A, 1	61	DB	—	—	—
ROL	A, 1	61	EB	—	—	—

Table 5 - 8 List of Instruction Formats (17/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
RORC	A, 1	61	FB	—	—	—
ROLC	A, 1	61	DC	—	—	—
ROLWC	AX, 1	61	EE	—	—	—
	BC, 1	61	FE	—	—	—
MOV1	CY, saddr.0	71	04	saddr	—	—
	CY, saddr.1	71	14	saddr	—	—
	CY, saddr.2	71	24	saddr	—	—
	CY, saddr.3	71	34	saddr	—	—
	CY, saddr.4	71	44	saddr	—	—
	CY, saddr.5	71	54	saddr	—	—
	CY, saddr.6	71	64	saddr	—	—
	CY, saddr.7	71	74	saddr	—	—
	CY, sfr.0	71	0C	sfr	—	—
	CY, sfr.1	71	1C	sfr	—	—
	CY, sfr.2	71	2C	sfr	—	—
	CY, sfr.3	71	3C	sfr	—	—
	CY, sfr.4	71	4C	sfr	—	—
	CY, sfr.5	71	5C	sfr	—	—
	CY, sfr.6	71	6C	sfr	—	—
	CY, sfr.7	71	7C	sfr	—	—
	CY, A.0	71	8C	—	—	—
	CY, A.1	71	9C	—	—	—
	CY, A.2	71	AC	—	—	—
	CY, A.3	71	BC	—	—	—
	CY, A.4	71	CC	—	—	—
	CY, A.5	71	DC	—	—	—
	CY, A.6	71	EC	—	—	—
	CY, A.7	71	FC	—	—	—
	CY, PSW.0	71	0C	FA	—	—
	CY, PSW.1	71	1C	FA	—	—
	CY, PSW.2	71	2C	FA	—	—
	CY, PSW.3	71	3C	FA	—	—
	CY, PSW.4	71	4C	FA	—	—
	CY, PSW.5	71	5C	FA	—	—
	CY, PSW.6	71	6C	FA	—	—
	CY, PSW.7	71	7C	FA	—	—
	CY, [HL].0	71	84	—	—	—
	CY, [HL].1	71	94	—	—	—
	CY, [HL].2	71	A4	—	—	—

Table 5 - 8 List of Instruction Formats (18/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOV1	CY, [HL].3	71	B4	—	—	—
	CY, [HL].4	71	C4	—	—	—
	CY, [HL].5	71	D4	—	—	—
	CY, [HL].6	71	E4	—	—	—
	CY, [HL].7	71	F4	—	—	—
	saddr.0, CY	71	01	saddr	—	—
	saddr.1, CY	71	11	saddr	—	—
	saddr.2, CY	71	21	saddr	—	—
	saddr.3, CY	71	31	saddr	—	—
	saddr.4, CY	71	41	saddr	—	—
	saddr.5, CY	71	51	saddr	—	—
	saddr.6, CY	71	61	saddr	—	—
	saddr.7, CY	71	71	saddr	—	—
	sfr.0, CY	71	09	sfr	—	—
	sfr.1, CY	71	19	sfr	—	—
	sfr.2, CY	71	29	sfr	—	—
	sfr.3, CY	71	39	sfr	—	—
	sfr.4, CY	71	49	sfr	—	—
	sfr.5, CY	71	59	sfr	—	—
	sfr.6, CY	71	69	sfr	—	—
	sfr.7, CY	71	79	sfr	—	—
	A.0, CY	71	89	—	—	—
	A.1, CY	71	99	—	—	—
	A.2, CY	71	A9	—	—	—
	A.3, CY	71	B9	—	—	—
	A.4, CY	71	C9	—	—	—
	A.5, CY	71	D9	—	—	—
	A.6, CY	71	E9	—	—	—
	A.7, CY	71	F9	—	—	—
	PSW.0, CY	71	09	FA	—	—
	PSW.1, CY	71	19	FA	—	—
	PSW.2, CY	71	29	FA	—	—
	PSW.3, CY	71	39	FA	—	—
	PSW.4, CY	71	49	FA	—	—
	PSW.5, CY	71	59	FA	—	—
	PSW.6, CY	71	69	FA	—	—
	PSW.7, CY	71	79	FA	—	—
	[HL].0, CY	71	81	—	—	—
	[HL].1, CY	71	91	—	—	—

Table 5 - 8 List of Instruction Formats (19/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOV1	[HL].2, CY	71	A1	—	—	—
	[HL].3, CY	71	B1	—	—	—
	[HL].4, CY	71	C1	—	—	—
	[HL].5, CY	71	D1	—	—	—
	[HL].6, CY	71	E1	—	—	—
	[HL].7, CY	71	F1	—	—	—
	CY, ES:[HL].0	11	71	84	—	—
	CY, ES:[HL].1	11	71	94	—	—
	CY, ES:[HL].2	11	71	A4	—	—
	CY, ES:[HL].3	11	71	B4	—	—
	CY, ES:[HL].4	11	71	C4	—	—
	CY, ES:[HL].5	11	71	D4	—	—
	CY, ES:[HL].6	11	71	E4	—	—
	CY, ES:[HL].7	11	71	F4	—	—
	ES:[HL].0, CY	11	71	81	—	—
	ES:[HL].1, CY	11	71	91	—	—
	ES:[HL].2, CY	11	71	A1	—	—
	ES:[HL].3, CY	11	71	B1	—	—
	ES:[HL].4, CY	11	71	C1	—	—
	ES:[HL].5, CY	11	71	D1	—	—
	ES:[HL].6, CY	11	71	E1	—	—
	ES:[HL].7, CY	11	71	F1	—	—
AND1	CY, saddr.0	71	05	saddr	—	—
	CY, saddr.1	71	15	saddr	—	—
	CY, saddr.2	71	25	saddr	—	—
	CY, saddr.3	71	35	saddr	—	—
	CY, saddr.4	71	45	saddr	—	—
	CY, saddr.5	71	55	saddr	—	—
	CY, saddr.6	71	65	saddr	—	—
	CY, saddr.7	71	75	saddr	—	—
	CY, sfr.0	71	0D	sfr	—	—
	CY, sfr.1	71	1D	sfr	—	—
	CY, sfr.2	71	2D	sfr	—	—
	CY, sfr.3	71	3D	sfr	—	—
	CY, sfr.4	71	4D	sfr	—	—
	CY, sfr.5	71	5D	sfr	—	—
	CY, sfr.6	71	6D	sfr	—	—
	CY, sfr.7	71	7D	sfr	—	—
	CY, A.0	71	8D	—	—	—

Table 5 - 8 List of Instruction Formats (20/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
AND1	CY, A.1	71	9D	—	—	—
	CY, A.2	71	AD	—	—	—
	CY, A.3	71	BD	—	—	—
	CY, A.4	71	CD	—	—	—
	CY, A.5	71	DD	—	—	—
	CY, A.6	71	ED	—	—	—
	CY, A.7	71	FD	—	—	—
	CY, PSW.0	71	0D	FA	—	—
	CY, PSW.1	71	1D	FA	—	—
	CY, PSW.2	71	2D	FA	—	—
	CY, PSW.3	71	3D	FA	—	—
	CY, PSW.4	71	4D	FA	—	—
	CY, PSW.5	71	5D	FA	—	—
	CY, PSW.6	71	6D	FA	—	—
	CY, PSW.7	71	7D	FA	—	—
	CY, [HL].0	71	85	—	—	—
	CY, [HL].1	71	95	—	—	—
	CY, [HL].2	71	A5	—	—	—
	CY, [HL].3	71	B5	—	—	—
	CY, [HL].4	71	C5	—	—	—
	CY, [HL].5	71	D5	—	—	—
	CY, [HL].6	71	E5	—	—	—
	CY, [HL].7	71	F5	—	—	—
	CY, ES:[HL].0	11	71	85	—	—
	CY, ES:[HL].1	11	71	95	—	—
	CY, ES:[HL].2	11	71	A5	—	—
	CY, ES:[HL].3	11	71	B5	—	—
	CY, ES:[HL].4	11	71	C5	—	—
	CY, ES:[HL].5	11	71	D5	—	—
	CY, ES:[HL].6	11	71	E5	—	—
	CY, ES:[HL].7	11	71	F5	—	—
OR1	CY, saddr.0	71	06	saddr	—	—
	CY, saddr.1	71	16	saddr	—	—
	CY, saddr.2	71	26	saddr	—	—
	CY, saddr.3	71	36	saddr	—	—
	CY, saddr.4	71	46	saddr	—	—
	CY, saddr.5	71	56	saddr	—	—
	CY, saddr.6	71	66	saddr	—	—
	CY, saddr.7	71	76	saddr	—	—

Table 5 - 8 List of Instruction Formats (21/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
OR1	CY, sfr.0	71	0E	sfr	—	—
	CY, sfr.1	71	1E	sfr	—	—
	CY, sfr.2	71	2E	sfr	—	—
	CY, sfr.3	71	3E	sfr	—	—
	CY, sfr.4	71	4E	sfr	—	—
	CY, sfr.5	71	5E	sfr	—	—
	CY, sfr.6	71	6E	sfr	—	—
	CY, sfr.7	71	7E	sfr	—	—
	CY, A.0	71	8E	—	—	—
	CY, A.1	71	9E	—	—	—
	CY, A.2	71	AE	—	—	—
	CY, A.3	71	BE	—	—	—
	CY, A.4	71	CE	—	—	—
	CY, A.5	71	DE	—	—	—
	CY, A.6	71	EE	—	—	—
	CY, A.7	71	FE	—	—	—
	CY, PSW.0	71	0E	FA	—	—
	CY, PSW.1	71	1E	FA	—	—
	CY, PSW.2	71	2E	FA	—	—
	CY, PSW.3	71	3E	FA	—	—
	CY, PSW.4	71	4E	FA	—	—
	CY, PSW.5	71	5E	FA	—	—
	CY, PSW.6	71	6E	FA	—	—
	CY, PSW.7	71	7E	FA	—	—
	CY, [HL].0	71	86	—	—	—
	CY, [HL].1	71	96	—	—	—
	CY, [HL].2	71	A6	—	—	—
	CY, [HL].3	71	B6	—	—	—
	CY, [HL].4	71	C6	—	—	—
	CY, [HL].5	71	D6	—	—	—
	CY, [HL].6	71	E6	—	—	—
	CY, [HL].7	71	F6	—	—	—
	CY, ES:[HL].0	11	71	86	—	—
	CY, ES:[HL].1	11	71	96	—	—
	CY, ES:[HL].2	11	71	A6	—	—
	CY, ES:[HL].3	11	71	B6	—	—
	CY, ES:[HL].4	11	71	C6	—	—
	CY, ES:[HL].5	11	71	D6	—	—
	CY, ES:[HL].6	11	71	E6	—	—

Table 5 - 8 List of Instruction Formats (22/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
OR1	CY, ES:[HL].7	11	71	F6	—	—
XOR1	CY, saddr.0	71	07	saddr	—	—
	CY, saddr.1	71	17	saddr	—	—
	CY, saddr.2	71	27	saddr	—	—
	CY, saddr.3	71	37	saddr	—	—
	CY, saddr.4	71	47	saddr	—	—
	CY, saddr.5	71	57	saddr	—	—
	CY, saddr.6	71	67	saddr	—	—
	CY, saddr.7	71	77	saddr	—	—
	CY, sfr.0	71	0F	sfr	—	—
	CY, sfr.1	71	1F	sfr	—	—
	CY, sfr.2	71	2F	sfr	—	—
	CY, sfr.3	71	3F	sfr	—	—
	CY, sfr.4	71	4F	sfr	—	—
	CY, sfr.5	71	5F	sfr	—	—
	CY, sfr.6	71	6F	sfr	—	—
	CY, sfr.7	71	7F	sfr	—	—
	CY, A.0	71	8F	—	—	—
	CY, A.1	71	9F	—	—	—
	CY, A.2	71	AF	—	—	—
	CY, A.3	71	BF	—	—	—
	CY, A.4	71	CF	—	—	—
	CY, A.5	71	DF	—	—	—
	CY, A.6	71	EF	—	—	—
	CY, A.7	71	FF	—	—	—
	CY, PSW.0	71	0F	FA	—	—
	CY, PSW.1	71	1F	FA	—	—
	CY, PSW.2	71	2F	FA	—	—
	CY, PSW.3	71	3F	FA	—	—
	CY, PSW.4	71	4F	FA	—	—
	CY, PSW.5	71	5F	FA	—	—
	CY, PSW.6	71	6F	FA	—	—
	CY, PSW.7	71	7F	FA	—	—
	CY, [HL].0	71	87	—	—	—
	CY, [HL].1	71	97	—	—	—
	CY, [HL].2	71	A7	—	—	—
	CY, [HL].3	71	B7	—	—	—
	CY, [HL].4	71	C7	—	—	—
	CY, [HL].5	71	D7	—	—	—

Table 5 - 8 List of Instruction Formats (23/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
XOR1	CY, [HL].6	71	E7	—	—	—
	CY, [HL].7	71	F7	—	—	—
	CY, ES:[HL].0	11	71	87	—	—
	CY, ES:[HL].1	11	71	97	—	—
	CY, ES:[HL].2	11	71	A7	—	—
	CY, ES:[HL].3	11	71	B7	—	—
	CY, ES:[HL].4	11	71	C7	—	—
	CY, ES:[HL].5	11	71	D7	—	—
	CY, ES:[HL].6	11	71	E7	—	—
	CY, ES:[HL].7	11	71	F7	—	—
SET1	saddr.0	71	02	saddr	—	—
	saddr.1	71	12	saddr	—	—
	saddr.2	71	22	saddr	—	—
	saddr.3	71	32	saddr	—	—
	saddr.4	71	42	saddr	—	—
	saddr.5	71	52	saddr	—	—
	saddr.6	71	62	saddr	—	—
	saddr.7	71	72	saddr	—	—
	sfr.0	71	0A	sfr	—	—
	sfr.1	71	1A	sfr	—	—
	sfr.2	71	2A	sfr	—	—
	sfr.3	71	3A	sfr	—	—
	sfr.4	71	4A	sfr	—	—
	sfr.5	71	5A	sfr	—	—
	sfr.6	71	6A	sfr	—	—
	sfr.7	71	7A	sfr	—	—
	A.0	71	8A	—	—	—
	A.1	71	9A	—	—	—
	A.2	71	AA	—	—	—
	A.3	71	BA	—	—	—
	A.4	71	CA	—	—	—
	A.5	71	DA	—	—	—
	A.6	71	EA	—	—	—
	A.7	71	FA	—	—	—
	!addr16.0	71	00	adrl	adrh	—
	!addr16.1	71	10	adrl	adrh	—
	!addr16.2	71	20	adrl	adrh	—
	!addr16.3	71	30	adrl	adrh	—
	!addr16.4	71	40	adrl	adrh	—

Table 5 - 8 List of Instruction Formats (24/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
SET1	!addr16.5	71	50	adrl	adrh	—
	!addr16.6	71	60	adrl	adrh	—
	!addr16.7	71	70	adrl	adrh	—
	PSW.0	71	0A	FA	—	—
	PSW.1	71	1A	FA	—	—
	PSW.2	71	2A	FA	—	—
	PSW.3	71	3A	FA	—	—
	PSW.4	71	4A	FA	—	—
	PSW.5	71	5A	FA	—	—
	PSW.6	71	6A	FA	—	—
	PSW.7	71	7A	FA	—	—
	[HL].0	71	82	—	—	—
	[HL].1	71	92	—	—	—
	[HL].2	71	A2	—	—	—
	[HL].3	71	B2	—	—	—
	[HL].4	71	C2	—	—	—
	[HL].5	71	D2	—	—	—
	[HL].6	71	E2	—	—	—
	[HL].7	71	F2	—	—	—
	ES:!addr16.0	11	71	00	adrl	adrh
	ES:!addr16.1	11	71	10	adrl	adrh
	ES:!addr16.2	11	71	20	adrl	adrh
	ES:!addr16.3	11	71	30	adrl	adrh
	ES:!addr16.4	11	71	40	adrl	adrh
	ES:!addr16.5	11	71	50	adrl	adrh
	ES:!addr16.6	11	71	60	adrl	adrh
	ES:!addr16.7	11	71	70	adrl	adrh
	ES:[HL].0	11	71	82	—	—
	ES:[HL].1	11	71	92	—	—
	ES:[HL].2	11	71	A2	—	—
	ES:[HL].3	11	71	B2	—	—
	ES:[HL].4	11	71	C2	—	—
	ES:[HL].5	11	71	D2	—	—
	ES:[HL].6	11	71	E2	—	—
	ES:[HL].7	11	71	F2	—	—
CLR1	saddr.0	71	03	saddr	—	—
	saddr.1	71	13	saddr	—	—
	saddr.2	71	23	saddr	—	—
	saddr.3	71	33	saddr	—	—

Table 5 - 8 List of Instruction Formats (25/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
CLR1	saddr.4	71	43	saddr	—	—
	saddr.5	71	53	saddr	—	—
	saddr.6	71	63	saddr	—	—
	saddr.7	71	73	saddr	—	—
	sfr.0	71	0B	sfr	—	—
	sfr.1	71	1B	sfr	—	—
	sfr.2	71	2B	sfr	—	—
	sfr.3	71	3B	sfr	—	—
	sfr.4	71	4B	sfr	—	—
	sfr.5	71	5B	sfr	—	—
	sfr.6	71	6B	sfr	—	—
	sfr.7	71	7B	sfr	—	—
	A.0	71	8B	—	—	—
	A.1	71	9B	—	—	—
	A.2	71	AB	—	—	—
	A.3	71	BB	—	—	—
	A.4	71	CB	—	—	—
	A.5	71	DB	—	—	—
	A.6	71	EB	—	—	—
	A.7	71	FB	—	—	—
	!addr16.0	71	08	adrl	adrh	—
	!addr16.1	71	18	adrl	adrh	—
	!addr16.2	71	28	adrl	adrh	—
	!addr16.3	71	38	adrl	adrh	—
	!addr16.4	71	48	adrl	adrh	—
	!addr16.5	71	58	adrl	adrh	—
	!addr16.6	71	68	adrl	adrh	—
	!addr16.7	71	78	adrl	adrh	—
	PSW.0	71	0B	FA	—	—
	PSW.1	71	1B	FA	—	—
	PSW.2	71	2B	FA	—	—
	PSW.3	71	3B	FA	—	—
	PSW.4	71	4B	FA	—	—
	PSW.5	71	5B	FA	—	—
	PSW.6	71	6B	FA	—	—
	PSW.7	71	7B	FA	—	—
	[HL].0	71	83	—	—	—
	[HL].1	71	93	—	—	—
	[HL].2	71	A3	—	—	—

Table 5 - 8 List of Instruction Formats (26/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
CLR1	[HL].3	71	B3	—	—	—
	[HL].4	71	C3	—	—	—
	[HL].5	71	D3	—	—	—
	[HL].6	71	E3	—	—	—
	[HL].7	71	F3	—	—	—
	ES:!addr16.0	11	71	08	adrl	adrh
	ES:!addr16.1	11	71	18	adrl	adrh
	ES:!addr16.2	11	71	28	adrl	adrh
	ES:!addr16.3	11	71	38	adrl	adrh
	ES:!addr16.4	11	71	48	adrl	adrh
	ES:!addr16.5	11	71	58	adrl	adrh
	ES:!addr16.6	11	71	68	adrl	adrh
	ES:!addr16.7	11	71	78	adrl	adrh
	ES:[HL].0	11	71	83	—	—
	ES:[HL].1	11	71	93	—	—
	ES:[HL].2	11	71	A3	—	—
	ES:[HL].3	11	71	B3	—	—
	ES:[HL].4	11	71	C3	—	—
	ES:[HL].5	11	71	D3	—	—
	ES:[HL].6	11	71	E3	—	—
	ES:[HL].7	11	71	F3	—	—
SET1	CY	71	80	—	—	—
CLR1	CY	71	88	—	—	—
NOT1	CY	71	C0	—	—	—
CALL	AX	61	CA	—	—	—
	BC	61	DA	—	—	—
	DE	61	EA	—	—	—
	HL	61	FA	—	—	—
	!addr20	FE	adrl	adrh	—	—
	!addr16	FD	adrl	adrh	—	—
	!!addr20	FC	adrl	adrh	adrs	—
CALLT	[0080h]	61	84	—	—	—
	[0082h]	61	94	—	—	—
	[0084h]	61	A4	—	—	—
	[0086h]	61	B4	—	—	—
	[0088h]	61	C4	—	—	—
	[008Ah]	61	D4	—	—	—
	[008Ch]	61	E4	—	—	—
	[008Eh]	61	F4	—	—	—

Table 5 - 8 List of Instruction Formats (27/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
CALLT	[0090h]	61	85	—	—	—
	[0092h]	61	95	—	—	—
	[0094h]	61	A5	—	—	—
	[0096h]	61	B5	—	—	—
	[0098h]	61	C5	—	—	—
	[009Ah]	61	D5	—	—	—
	[009Ch]	61	E5	—	—	—
	[009Eh]	61	F5	—	—	—
	[00A0h]	61	86	—	—	—
	[00A2h]	61	96	—	—	—
	[00A4h]	61	A6	—	—	—
	[00A6h]	61	B6	—	—	—
	[00A8h]	61	C6	—	—	—
	[00AAh]	61	D6	—	—	—
	[00ACh]	61	E6	—	—	—
	[00AEh]	61	F6	—	—	—
	[00B0h]	61	87	—	—	—
	[00B2h]	61	97	—	—	—
	[00B4h]	61	A7	—	—	—
	[00B6h]	61	B7	—	—	—
	[00B8h]	61	C7	—	—	—
	[00BAh]	61	D7	—	—	—
	[00BCh]	61	E7	—	—	—
	[00BEh]	61	F7	—	—	—
BRK	—	61	CC	—	—	—
RET	—	D7	—	—	—	—
RETI	—	61	FC	—	—	—
RETB	—	61	EC	—	—	—
PUSH	PSW	61	DD	—	—	—
	AX	C1	—	—	—	—
	BC	C3	—	—	—	—
	DE	C5	—	—	—	—
	HL	C7	—	—	—	—
POP	PSW	61	CD	—	—	—
	AX	C0	—	—	—	—
	BC	C2	—	—	—	—
	DE	C4	—	—	—	—
	HL	C6	—	—	—	—

Table 5 - 8 List of Instruction Formats (28/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
MOVW	SP, #word	CB	F8	datal	datah	—
	SP, AX	BE	F8	—	—	—
	AX, SP	AE	F8	—	—	—
	BC, SP	DB	adrl	adrh	—	—
	DE, SP	EB	adrl	adrh	—	—
	HL, SP	FB	adrl	adrh	—	—
ADDW	SP, #byte	10	data	—	—	—
SUBW	SP, #byte	20	data	—	—	—
BR	AX	61	CB	—	—	—
	\$addr20	EF	adr	—	—	—
	\$!addr20	EE	adrl	adrh	—	—
	!addr16	ED	adrl	adrh	—	—
	!!addr20	EC	adrl	adrh	adrs	—
BC	\$addr20	DC	adr	—	—	—
BNC	\$addr20	DE	adr	—	—	—
BZ	\$addr20	DD	adr	—	—	—
BNZ	\$addr20	DF	adr	—	—	—
BH	\$addr20	61	C3	adr	—	—
BNH	\$addr20	61	D3	adr	—	—
BT	saddr.0, \$addr20	31	02	saddr	adr	—
	saddr.1, \$addr20	31	12	saddr	adr	—
	saddr.2, \$addr20	31	22	saddr	adr	—
	saddr.3, \$addr20	31	32	saddr	adr	—
	saddr.4, \$addr20	31	42	saddr	adr	—
	saddr.5, \$addr20	31	52	saddr	adr	—
	saddr.6, \$addr20	31	62	saddr	adr	—
	saddr.7, \$addr20	31	72	saddr	adr	—
	sfr.0, \$addr20	31	82	sfr	adr	—
	sfr.1, \$addr20	31	92	sfr	adr	—
	sfr.2, \$addr20	31	A2	sfr	adr	—
	sfr.3, \$addr20	31	B2	sfr	adr	—
	sfr.4, \$addr20	31	C2	sfr	adr	—
	sfr.5, \$addr20	31	D2	sfr	adr	—
	sfr.6, \$addr20	31	E2	sfr	adr	—
	sfr.7, \$addr20	31	F2	sfr	adr	—
	A.0, \$addr20	31	03	adr	—	—
	A.1, \$addr20	31	13	adr	—	—
	A.2, \$addr20	31	23	adr	—	—
	A.3, \$addr20	31	33	adr	—	—

Table 5 - 8 List of Instruction Formats (29/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
BT	A.4, \$addr20	31	43	adr	—	—
	A.5, \$addr20	31	53	adr	—	—
	A.6, \$addr20	31	63	adr	—	—
	A.7, \$addr20	31	73	adr	—	—
	PSW.0, \$addr20	31	82	FA	adr	—
	PSW.1, \$addr20	31	92	FA	adr	—
	PSW.2, \$addr20	31	A2	FA	adr	—
	PSW.3, \$addr20	31	B2	FA	adr	—
	PSW.4, \$addr20	31	C2	FA	adr	—
	PSW.5, \$addr20	31	D2	FA	adr	—
	PSW.6, \$addr20	31	E2	FA	adr	—
	PSW.7, \$addr20	31	F2	FA	adr	—
	[HL].0, \$addr20	31	83	adr	—	—
	[HL].1, \$addr20	31	93	adr	—	—
	[HL].2, \$addr20	31	A3	adr	—	—
	[HL].3, \$addr20	31	B3	adr	—	—
	[HL].4, \$addr20	31	C3	adr	—	—
	[HL].5, \$addr20	31	D3	adr	—	—
	[HL].6, \$addr20	31	E3	adr	—	—
	[HL].7, \$addr20	31	F3	adr	—	—
	ES:[HL].0, \$addr20	11	31	83	adr	—
	ES:[HL].1, \$addr20	11	31	93	adr	—
	ES:[HL].2, \$addr20	11	31	A3	adr	—
	ES:[HL].3, \$addr20	11	31	B3	adr	—
	ES:[HL].4, \$addr20	11	31	C3	adr	—
	ES:[HL].5, \$addr20	11	31	D3	adr	—
	ES:[HL].6, \$addr20	11	31	E3	adr	—
	ES:[HL].7, \$addr20	11	31	F3	adr	—
BF	saddr.0, \$addr20	31	04	saddr	adr	—
	saddr.1, \$addr20	31	14	saddr	adr	—
	saddr.2, \$addr20	31	24	saddr	adr	—
	saddr.3, \$addr20	31	34	saddr	adr	—
	saddr.4, \$addr20	31	44	saddr	adr	—
	saddr.5, \$addr20	31	54	saddr	adr	—
	saddr.6, \$addr20	31	64	saddr	adr	—
	saddr.7, \$addr20	31	74	saddr	adr	—
	sfr.0, \$addr20	31	84	sfr	adr	—
	sfr.1, \$addr20	31	94	sfr	adr	—
	sfr.2, \$addr20	31	A4	sfr	adr	—

Table 5 - 8 List of Instruction Formats (30/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
BF	sfr.3,\$addr20	31	B4	sfr	adr	—
	sfr.4,\$addr20	31	C4	sfr	adr	—
	sfr.5,\$addr20	31	D4	sfr	adr	—
	sfr.6,\$addr20	31	E4	sfr	adr	—
	sfr.7,\$addr20	31	F4	sfr	adr	—
	A.0,\$addr20	31	05	adr	—	—
	A.1,\$addr20	31	15	adr	—	—
	A.2,\$addr20	31	25	adr	—	—
	A.3,\$addr20	31	35	adr	—	—
	A.4,\$addr20	31	45	adr	—	—
	A.5,\$addr20	31	55	adr	—	—
	A.6,\$addr20	31	65	adr	—	—
	A.7,\$addr20	31	75	adr	—	—
	PSW.0,\$addr20	31	84	FA	adr	—
	PSW.1,\$addr20	31	94	FA	adr	—
	PSW.2,\$addr20	31	A4	FA	adr	—
	PSW.3,\$addr20	31	B4	FA	adr	—
	PSW.4,\$addr20	31	C4	FA	adr	—
	PSW.5,\$addr20	31	D4	FA	adr	—
	PSW.6,\$addr20	31	E4	FA	adr	—
	PSW.7,\$addr20	31	F4	FA	adr	—
	[HL].0,\$addr20	31	85	adr	—	—
	[HL].1,\$addr20	31	95	adr	—	—
	[HL].2,\$addr20	31	A5	adr	—	—
	[HL].3,\$addr20	31	B5	adr	—	—
	[HL].4,\$addr20	31	C5	adr	—	—
	[HL].5,\$addr20	31	D5	adr	—	—
	[HL].6,\$addr20	31	E5	adr	—	—
	[HL].7,\$addr20	31	F5	adr	—	—
	ES:[HL].0,\$addr20	11	31	85	adr	—
	ES:[HL].1,\$addr20	11	31	95	adr	—
	ES:[HL].2,\$addr20	11	31	A5	adr	—
	ES:[HL].3,\$addr20	11	31	B5	adr	—
	ES:[HL].4,\$addr20	11	31	C5	adr	—
	ES:[HL].5,\$addr20	11	31	D5	adr	—
	ES:[HL].6,\$addr20	11	31	E5	adr	—
	ES:[HL].7,\$addr20	11	31	F5	adr	—

Table 5 - 8 List of Instruction Formats (31/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
BTCLR	saddr.0, \$addr20	31	00	saddr	adr	—
	saddr.1, \$addr20	31	10	saddr	adr	—
	saddr.2, \$addr20	31	20	saddr	adr	—
	saddr.3, \$addr20	31	30	saddr	adr	—
	saddr.4, \$addr20	31	40	saddr	adr	—
	saddr.5, \$addr20	31	50	saddr	adr	—
	saddr.6, \$addr20	31	60	saddr	adr	—
	saddr.7, \$addr20	31	70	saddr	adr	—
	sfr.0, \$addr20	31	80	sfr	adr	—
	sfr.1, \$addr20	31	90	sfr	adr	—
	sfr.2, \$addr20	31	A0	sfr	adr	—
	sfr.3, \$addr20	31	B0	sfr	adr	—
	sfr.4, \$addr20	31	C0	sfr	adr	—
	sfr.5, \$addr20	31	D0	sfr	adr	—
	sfr.6, \$addr20	31	E0	sfr	adr	—
	sfr.7, \$addr20	31	F0	sfr	adr	—
	A.0, \$addr20	31	01	adr	—	—
	A.1, \$addr20	31	11	adr	—	—
	A.2, \$addr20	31	21	adr	—	—
	A.3, \$addr20	31	31	adr	—	—
	A.4, \$addr20	31	41	adr	—	—
	A.5, \$addr20	31	51	adr	—	—
	A.6, \$addr20	31	61	adr	—	—
	A.7, \$addr20	31	71	adr	—	—
	PSW.0, \$addr20	31	80	FA	adr	—
	PSW.1, \$addr20	31	90	FA	adr	—
	PSW.2, \$addr20	31	A0	FA	adr	—
	PSW.3, \$addr20	31	B0	FA	adr	—
	PSW.4, \$addr20	31	C0	FA	adr	—
	PSW.5, \$addr20	31	D0	FA	adr	—
	PSW.6, \$addr20	31	E0	FA	adr	—
	PSW.7, \$addr20	31	F0	FA	adr	—
	[HL].0, \$addr20	31	81	adr	—	—
	[HL].1, \$addr20	31	91	adr	—	—
	[HL].2, \$addr20	31	A1	adr	—	—
	[HL].3, \$addr20	31	B1	adr	—	—
	[HL].4, \$addr20	31	C1	adr	—	—
	[HL].5, \$addr20	31	D1	adr	—	—
	[HL].6, \$addr20	31	E1	adr	—	—

Table 5 - 8 List of Instruction Formats (32/32)

Mnemonic	Operands	Opcode				
		1st	2nd	3rd	4th	5th
BTCLR	[HL].7, \$addr20	31	F1	adr	—	—
	ES:[HL].0, \$addr20	11	31	81	adr	—
	ES:[HL].1, \$addr20	11	31	91	adr	—
	ES:[HL].2, \$addr20	11	31	A1	adr	—
	ES:[HL].3, \$addr20	11	31	B1	adr	—
	ES:[HL].4, \$addr20	11	31	C1	adr	—
	ES:[HL].5, \$addr20	11	31	D1	adr	—
	ES:[HL].6, \$addr20	11	31	E1	adr	—
	ES:[HL].7, \$addr20	11	31	F1	adr	—
SKC	—	61	C8	—	—	—
SKNC	—	61	D8	—	—	—
SKZ	—	61	E8	—	—	—
SKNZ	—	61	F8	—	—	—
SKH	—	61	E3	—	—	—
SKNH	—	61	F3	—	—	—
SEL Note 2	RB0	61	CF	—	—	—
	RB1	61	DF	—	—	—
	RB2	61	EF	—	—	—
	RB3	61	FF	—	—	—
NOP	—	00	—	—	—	—
EI	—	71	7A	FA	—	—
DI	—	71	7B	FA	—	—
HALT	—	61	ED	—	—	—
STOP	—	61	FD	—	—	—
PREFIX	—	11	—	—	—	—

Note 1. Only the instruction set of the RL78-S3 core includes this extended instruction.

Note 2. The instruction set of the RL78-S1 core does not include this instruction.

5.7 Instruction Maps

Tables 5 - 9 to 5 - 12 show instruction maps.

Table 5 - 9 Instruction Map (1st MAP)

	0 (low)	1 (low)	2 (low)	3 (low)	4 (low)	5 (low)	6 (low)	7 (low)	8 (low)	9 (low)	a (low)	b (low)	c (low)	d (low)	e (low)	f (low)
0	NOP	ADDW AX, AX	ADDW AX, laddr16	ADDW AX, BC	ADDW AX, #word	ADDW AX, DE	ADDW AX, saddrp	ADDW AX, HL	XCH X, X	MOV A, word[B]	ADD saddr, #byte	ADD A, saddr	ADD A, #byte	ADD A, [HL]	ADD A, [HL+byte]	ADD A, laddr16
1	ADDW SP, #byte	PREFIX	MOVW BC, AX	MOVW AX, BC	MOVW DE, AX	MOVW AX, DE	MOVW HL, AX	MOVW AX, HL	MOV word[C], A	MOV word[B], #byte	ADDC saddr, #byte	ADDC A, saddr	ADDC A, #byte	ADDC A, [HL]	ADDC A, [HL+byte]	ADDC A, laddr16
2	SUBW SP, #byte		SUBW AX, laddr16	SUBW AX, BC	SUBW AX, #word	SUBW AX, DE	SUBW AX, saddrp	SUBW AX, HL	MOV word[C], A	MOV A, word[C]	SUB saddr, #byte	SUB A, saddr	SUB A, #byte	SUB A, [HL]	SUB A, [HL+byte]	SUB A, laddr16
3	MOVW AX, #word	4th MAP	MOVW BC, #word	XCHW AX, BC	MOVW DE, #word	XCHW AX, DE	MOVW HL, #word	XCHW AX, HL	MOV word[C], #byte	MOV word[BC], #byte	SUBC saddr, #byte	SUBC A, saddr	SUBC A, #byte	SUBC A, [HL]	SUBC A, [HL+byte]	SUBC A, laddr16
4	CMP laddr16, #byte	MOV ES, #byte	CMPW AX, laddr16	CMPW AX, BC	CMPW AX, #word	CMPW AX, DE	CMPW AX, saddrp	CMPW AX, HL	MOV word[BC], A	MOV A, word[BC]	CMP saddr, #byte	CMP A, saddr	CMP A, #byte	CMP A, [HL]	CMP A, [HL+byte]	CMP A, laddr16
5	MOV X, #byte	MOV A, #byte	MOV C, #byte	MOV B, #byte	MOV E, #byte	MOV D, #byte	MOV L, #byte	MOV H, #byte	MOVW word[B], AX	MOVW AX, word[B]	AND saddr, #byte	AND A, saddr	AND A, #byte	AND A, [HL]	AND A, [HL+byte]	AND A, laddr16
6	MOV A, X	2nd MAP	MOV A, C	MOV A, B	MOV A, E	MOV A, D	MOV A, L	MOV A, H	MOVW word[C], AX	MOVW AX, word[C]	OR saddr, #byte	OR A, saddr	OR A, #byte	OR A, [HL]	OR A, [HL+byte]	OR A, laddr16
7	MOV X, A	3rd MAP	MOV C, A	MOV B, A	MOV E, A	MOV D, A	MOV L, A	MOV H, A	MOVW word[BC], AX	MOVW AX, word[BC]	XOR saddr, #byte	XOR A, saddr	XOR A, #byte	XOR A, [HL]	XOR A, [HL+byte]	XOR A, laddr16
8	INC X	INC A	INC C	INC B	INC E	INC D	INC L	INC H	MOV A, [SP+byte]	MOV A, [DE]	MOV A, [DE+byte]	MOV A, [HL]	MOV A, [HL+byte]	MOV A, saddr	MOV A, sfr	MOV A, laddr16
9	DEC X	DEC A	DEC C	DEC B	DEC E	DEC D	DEC L	DEC H	MOV [SP+byte], A	MOV [DE], A	[DE+byte], A	[HL], A	[HL+byte], A	MOV saddr, A	MOV sfr, A	MOV laddr16, A
a	INC laddr16	INCW AX	INCW laddr16	INCW BC	INC saddr	INCW DE	INCW saddrp	INCW HL	MOVW AX, [SP+byte]	MOVW AX, [DE]	AX, [DE+byte]	MOVW AX, [HL]	MOVW AX, [HL+byte]	MOVW AX, saddrp	MOVW AX, sfrp	MOVW AX, laddr16
b	DEC laddr16	DECW AX	DECW laddr16	DECW BC	DEC saddr	DECW DE	DECW saddrp	DECW HL	MOVW [SP+byte], AX	MOVW [DE], AX	[DE+byte], AX	MOVW [HL], AX	MOVW [HL+byte], AX	MOVW saddrp, AX	MOVW sfrp, AX	MOVW laddr16, AX
c	POP AX	PUSH AX	POP BC	PUSH BC	POP DE	PUSH DE	POP HL	PUSH HL	MOV [SP+byte], #byte	MOVW saddrp, #word	MOV [DE+byte], #byte	MOVW sfrp, #word	MOV [HL+byte], #byte	MOV saddr, #byte	NOTEMOV sfr, #byte	MOV laddr16, #byte
d	CMP0 X	CMP0 A	CMP0 C	CMP0 B	CMP0 saddr	CMP0 laddr16	MULU X	RET	MOV X, saddr	MOV X, laddr16	MOVW BC, saddrp	MOVW BC, laddr16	BC \$saddr20	BZ \$saddr20	BNC \$saddr20	BNZ \$saddr20
e	ONEB X	ONEB A	ONEB C	ONEB B	ONEB saddr	ONEB laddr16	ONEW AX	ONEW BC	MOV B, saddr	MOV B, laddr16	DE, saddrp	MOVW DE, laddr16	BR laddr20	BR laddr16	BR \$laddr20	BR \$saddr20
f	CLRB X	CLRB A	CLRB C	CLRB B	CLRB saddr	CLRB laddr16	CLRW AX	CLRW BC	MOV C, saddr	MOV C, laddr16	HL, saddrp	MOVW HL, laddr16	CALL laddr20	CALL laddr16	CALL \$laddr20	

Note From among the multiply, divide, and multiply-and-accumulate instructions, MULHU, MULH, DIVHU, DIVWU, MACHU, and MACH are also mapped here.

Table 5 - 10 Instruction Map (2nd MAP)

	0 (low)	1 (low)	2 (low)	3 (low)	4 (low)	5 (low)	6 (low)	7 (low)	8 (low)	9 (low)	a (low)	b (low)	c (low)	d (low)	e (low)	f (low)
0	ADD X, A	ADD A, A	ADD C, A	ADD B, A	ADD E, A	ADD D, A	ADD L, A	ADD H, A	ADD A, X	ADDW AX, [HL+byte]	ADD A, C	ADD A, B	ADD A, E	ADD A, D	ADD A, L	ADD A, H
1	ADDC X, A	ADDC A, A	ADDC C, A	ADDC B, A	ADDC E, A	ADDC D, A	ADDC L, A	ADDC H, A	ADDC A, X		ADDC A, C	ADDC A, B	ADDC A, E	ADDC A, D	ADDC A, L	ADDC A, H
2	SUB X, A	SUB A, A	SUB C, A	SUB B, A	SUB E, A	SUB D, A	SUB L, A	SUB H, A	SUB A, X	SUBW AX, [HL+byte]	SUB A, C	SUB A, B	SUB A, E	SUB A, D	SUB A, L	SUB A, H
3	SUBC X, A	SUBC A, A	SUBC C, A	SUBC B, A	SUBC E, A	SUBC D, A	SUBC L, A	SUBC H, A	SUBC A, X		SUBC A, C	SUBC A, B	SUBC A, E	SUBC A, D	SUBC A, L	SUBC A, H
4	CMP X, A	CMP A, A	CMP C, A	CMP B, A	CMP E, A	CMP D, A	CMP L, A	CMP H, A	CMP A, X	CMPW AX, [HL+byte]	CMP A, C	CMP A, B	CMP A, E	CMP A, D	CMP A, L	CMP A, H
5	AND X, A	AND A, A	AND C, A	AND B, A	AND E, A	AND D, A	AND L, A	AND H, A	AND A, X	INC [HL+byte]	AND A, C	AND A, B	AND A, E	AND A, D	AND A, L	AND A, H
6	OR X, A	OR A, A	OR C, A	OR B, A	OR E, A	OR D, A	OR L, A	OR H, A	OR A, X	DEC [HL+byte]	OR A, C	OR A, B	OR A, E	OR A, D	OR A, L	OR A, H
7	XOR X, A	XOR A, A	XOR C, A	XOR B, A	XOR E, A	XOR D, A	XOR L, A	XOR H, A	XOR A, X	INCW [HL+byte]	XOR A, C	XOR A, B	XOR A, E	XOR A, D	XOR A, L	XOR A, H
8	ADD A, [HL+B]		ADD A, [HL+C]		CALLT [0080h]	CALLT [0090h]	CALLT [00A0h]	CALLT [00B0h]		DECW [HL+byte]	XCH A, C	XCH A, B	XCH A, E	XCH A, D	XCH A, L	XCH A, H
9	ADDC A, [HL+B]		ADDC A, [HL+C]		CALLT [0082h]	CALLT [0092h]	CALLT [00A2h]	CALLT [00B2h]								
a	SUB A, [HL+B]		SUB A, [HL+C]		CALLT [0084h]	CALLT [0094h]	CALLT [00A4h]	CALLT [00B4h]	XCH A, saddr	XCH A, [HL+C]	XCH A, faddr16	XCH A, sir	XCH A, [HL]	XCH A, [HL+byte]	XCH A, [DE]	XCH A, [DE+byte]
b	SUBC A, [HL+B]		SUBC A, [HL+C]		CALLT [0086h]	CALLT [0096h]	CALLT [00A6h]	CALLT [00B6h]	MOV ES, saddr	XCH A, [HL+B]						
c	CMP A, [HL+B]		CMP A, [HL+C]	BH \$addr20	CALLT [0088h]	CALLT [0098h]	CALLT [00A8h]	CALLT [00B8h]	SKC	MOV A, [HL+B]	CALL AX	BR AX	BRK	POP PSW	MOVSW [HL+byte], X	SEL RB0Note
d	AND A, [HL+B]		AND A, [HL+C]	BNH \$addr20	CALLT [008Ah]	CALLT [009Ah]	CALLT [00AAh]	CALLT [00BAh]	SKNC	MOV [HL+B], A	CALL BC	ROR A, 1	ROL A, 1	PUSH PSW	CMPS X, [HL+byte]	SEL RB1Note
e	OR A, [HL+B]		OR A, [HL+C]	SKH	CALLT [008Ch]	CALLT [009Ch]	CALLT [00ACH]	CALLT [00BCh]	SKZ	MOV A, [HL+C]	CALL DE	ROL A, 1	RETB	HALT	ROLWC AX, 1	SEL RB2Note
f	XOR A, [HL+B]		XOR A, [HL+C]	SKNH	CALLT [008Eh]	CALLT [009Eh]	CALLT [00AEh]	CALLT [00BEh]	SKNZ	MOV [HL+C], A	CALL HL	RORC A, 1	RETI	STOP	ROLWC BC, 1	SEL RB3Note

Note The instruction set of the RL78-S1 core does not include this instruction.

Table 5 - 11 Instruction Map (3rd MAP)

	0 (low)	1 (low)	2 (low)	3 (low)	4 (low)	5 (low)	6 (low)	7 (low)	8 (low)	9 (low)	a (low)	b (low)	c (low)	d (low)	e (low)	f (low)
0	SET1 laddr16.0	MOV1 saddr.0, CY	SET1 saddr.0	CLR1 saddr.0	MOV1 CY, saddr.0	AND1 CY, saddr.0	OR1 CY, saddr.0	XOR1 CY, saddr.0	CLR1 laddr16.0	MOV1 sfr.0, CY	SET1 sfr.0	CLR1 sfr.0	MOV1 CY, sfr.0	AND1 CY, sfr.0	OR1 CY, sfr.0	XOR1 CY, sfr.0
1	SET1 laddr16.1	MOV1 saddr.1, CY	SET1 saddr.1	CLR1 saddr.1	MOV1 CY, saddr.1	AND1 CY, saddr.1	OR1 CY, saddr.1	XOR1 CY, saddr.1	CLR1 laddr16.1	MOV1 sfr.1, CY	SET1 sfr.1	CLR1 sfr.1	MOV1 CY, sfr.1	AND1 CY, sfr.1	OR1 CY, sfr.1	XOR1 CY, sfr.1
2	SET1 laddr16.2	MOV1 saddr.2, CY	SET1 saddr.2	CLR1 saddr.2	MOV1 CY, saddr.2	AND1 CY, saddr.2	OR1 CY, saddr.2	XOR1 CY, saddr.2	CLR1 laddr16.2	MOV1 sfr.2, CY	SET1 sfr.2	CLR1 sfr.2	MOV1 CY, sfr.2	AND1 CY, sfr.2	OR1 CY, sfr.2	XOR1 CY, sfr.2
3	SET1 laddr16.3	MOV1 saddr.3, CY	SET1 saddr.3	CLR1 saddr.3	MOV1 CY, saddr.3	AND1 CY, saddr.3	OR1 CY, saddr.3	XOR1 CY, saddr.3	CLR1 laddr16.3	MOV1 sfr.3, CY	SET1 sfr.3	CLR1 sfr.3	MOV1 CY, sfr.3	AND1 CY, sfr.3	OR1 CY, sfr.3	XOR1 CY, sfr.3
4	SET1 laddr16.4	MOV1 saddr.4, CY	SET1 saddr.4	CLR1 saddr.4	MOV1 CY, saddr.4	AND1 CY, saddr.4	OR1 CY, saddr.4	XOR1 CY, saddr.4	CLR1 laddr16.4	MOV1 sfr.4, CY	SET1 sfr.4	CLR1 sfr.4	MOV1 CY, sfr.4	AND1 CY, sfr.4	OR1 CY, sfr.4	XOR1 CY, sfr.4
5	SET1 laddr16.5	MOV1 saddr.5, CY	SET1 saddr.5	CLR1 saddr.5	MOV1 CY, saddr.5	AND1 CY, saddr.5	OR1 CY, saddr.5	XOR1 CY, saddr.5	CLR1 laddr16.5	MOV1 sfr.5, CY	SET1 sfr.5	CLR1 sfr.5	MOV1 CY, sfr.5	AND1 CY, sfr.5	OR1 CY, sfr.5	XOR1 CY, sfr.5
6	SET1 laddr16.6	MOV1 saddr.6, CY	SET1 saddr.6	CLR1 saddr.6	MOV1 CY, saddr.6	AND1 CY, saddr.6	OR1 CY, saddr.6	XOR1 CY, saddr.6	CLR1 laddr16.6	MOV1 sfr.6, CY	SET1 sfr.6	CLR1 sfr.6	MOV1 CY, sfr.6	AND1 CY, sfr.6	OR1 CY, sfr.6	XOR1 CY, sfr.6
7	SET1 laddr16.7	MOV1 saddr.7, CY	SET1 saddr.7	CLR1 saddr.7	MOV1 CY, saddr.7	AND1 CY, saddr.7	OR1 CY, saddr.7	XOR1 CY, saddr.7	CLR1 laddr16.7	MOV1 sfr.7, CY	SET1 sfr.7	CLR1 sfr.7	MOV1 CY, sfr.7	AND1 CY, sfr.7	OR1 CY, sfr.7	XOR1 CY, sfr.7
8	SET1 CY	MOV1 [HL], 0, CY	SET1 [HL], 0	CLR1 [HL], 0	MOV1 CY, [HL], 0	AND1 CY, [HL], 0	OR1 CY, [HL], 0	XOR1 CY, [HL], 0	CLR1 CY	MOV1 A.0, CY	SET1 A.0	CLR1 A.0	MOV1 CY, A.0	AND1 CY, A.0	OR1 CY, A.0	XOR1 CY, A.0
9		MOV1 [HL], 1, CY	SET1 [HL], 1	CLR1 [HL], 1	MOV1 CY, [HL], 1	AND1 CY, [HL], 1	OR1 CY, [HL], 1	XOR1 CY, [HL], 1		MOV1 A.1, CY	SET1 A.1	CLR1 A.1	MOV1 CY, A.1	AND1 CY, A.1	OR1 CY, A.1	XOR1 CY, A.1
a		MOV1 [HL], 2, CY	SET1 [HL], 2	CLR1 [HL], 2	MOV1 CY, [HL], 2	AND1 CY, [HL], 2	OR1 CY, [HL], 2	XOR1 CY, [HL], 2		MOV1 A.2, CY	SET1 A.2	CLR1 A.2	MOV1 CY, A.2	AND1 CY, A.2	OR1 CY, A.2	XOR1 CY, A.2
b		MOV1 [HL], 3, CY	SET1 [HL], 3	CLR1 [HL], 3	MOV1 CY, [HL], 3	AND1 CY, [HL], 3	OR1 CY, [HL], 3	XOR1 CY, [HL], 3		MOV1 A.3, CY	SET1 A.3	CLR1 A.3	MOV1 CY, A.3	AND1 CY, A.3	OR1 CY, A.3	XOR1 CY, A.3
c	NOT1 CY	MOV1 [HL], 4, CY	SET1 [HL], 4	CLR1 [HL], 4	MOV1 CY, [HL], 4	AND1 CY, [HL], 4	OR1 CY, [HL], 4	XOR1 CY, [HL], 4		MOV1 A.4, CY	SET1 A.4	CLR1 A.4	MOV1 CY, A.4	AND1 CY, A.4	OR1 CY, A.4	XOR1 CY, A.4
d		MOV1 [HL], 5, CY	SET1 [HL], 5	CLR1 [HL], 5	MOV1 CY, [HL], 5	AND1 CY, [HL], 5	OR1 CY, [HL], 5	XOR1 CY, [HL], 5		MOV1 A.5, CY	SET1 A.5	CLR1 A.5	MOV1 CY, A.5	AND1 CY, A.5	OR1 CY, A.5	XOR1 CY, A.5
e		MOV1 [HL], 6, CY	SET1 [HL], 6	CLR1 [HL], 6	MOV1 CY, [HL], 6	AND1 CY, [HL], 6	OR1 CY, [HL], 6	XOR1 CY, [HL], 6		MOV1 A.6, CY	SET1 A.6	CLR1 A.6	MOV1 CY, A.6	AND1 CY, A.6	OR1 CY, A.6	XOR1 CY, A.6
f		MOV1 [HL], 7, CY	SET1 [HL], 7	CLR1 [HL], 7	MOV1 CY, [HL], 7	AND1 CY, [HL], 7	OR1 CY, [HL], 7	XOR1 CY, [HL], 7		MOV1 A.7, CY	SET1 A.7	CLR1 A.7	MOV1 CY, A.7	AND1 CY, A.7	OR1 CY, A.7	XOR1 CY, A.7

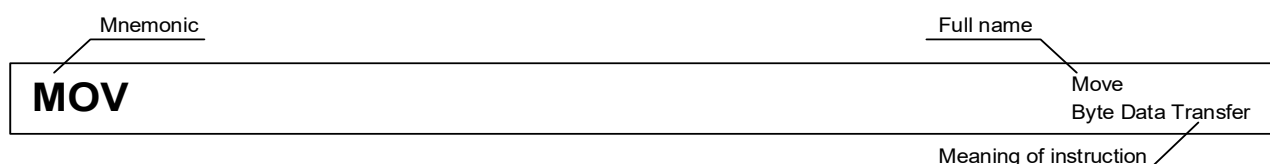
Table 5 - 12 Instruction Map (4th MAP)

	0 (low)	1 (low)	2 (low)	3 (low)	4 (low)	5 (low)	6 (low)	7 (low)	8 (low)	9 (low)	a (low)	b (low)	c (low)	d (low)	e (low)	f (low)
0	BTCLR saddr.0, \$addr20	BTCLR A.0, \$addr20	BT saddr.0, \$addr20	BT A.0, \$addr20	BF saddr.0, \$addr20	BF A.0, \$addr20										
1	BTCLR saddr.1, \$addr20	BTCLR A.1, \$addr20	BT saddr.1, \$addr20	BT A.1, \$addr20	BF saddr.1, \$addr20	BF A.1, \$addr20		SHL C, 1	SHL B, 1	SHL A, 1	SHR A, 1	SAR A, 1	SHLW BC, 1	SHLW AX, 1	SHRW AX, 1	SARW AX, 1
2	BTCLR saddr.2, \$addr20	BTCLR A.2, \$addr20	BT saddr.2, \$addr20	BT A.2, \$addr20	BF saddr.2, \$addr20	BF A.2, \$addr20		SHL C, 2	SHL B, 2	SHL A, 2	SHR A, 2	SAR A, 2	SHLW BC, 2	SHLW AX, 2	SHRW AX, 2	SARW AX, 2
3	BTCLR saddr.3, \$addr20	BTCLR A.3, \$addr20	BT saddr.3, \$addr20	BT A.3, \$addr20	BF saddr.3, \$addr20	BF A.3, \$addr20		SHL C, 3	SHL B, 3	SHL A, 3	SHR A, 3	SAR A, 3	SHLW BC, 3	SHLW AX, 3	SHRW AX, 3	SARW AX, 3
4	BTCLR saddr.4, \$addr20	BTCLR A.4, \$addr20	BT saddr.4, \$addr20	BT A.4, \$addr20	BF saddr.4, \$addr20	BF A.4, \$addr20		SHL C, 4	SHL B, 4	SHL A, 4	SHR A, 4	SAR A, 4	SHLW BC, 4	SHLW AX, 4	SHRW AX, 4	SARW AX, 4
5	BTCLR saddr.5, \$addr20	BTCLR A.5, \$addr20	BT saddr.5, \$addr20	BT A.5, \$addr20	BF saddr.5, \$addr20	BF A.5, \$addr20		SHL C, 5	SHL B, 5	SHL A, 5	SHR A, 5	SAR A, 5	SHLW BC, 5	SHLW AX, 5	SHRW AX, 5	SARW AX, 5
6	BTCLR saddr.6, \$addr20	BTCLR A.6, \$addr20	BT saddr.6, \$addr20	BT A.6, \$addr20	BF saddr.6, \$addr20	BF A.6, \$addr20		SHL C, 6	SHL B, 6	SHL A, 6	SHR A, 6	SAR A, 6	SHLW BC, 6	SHLW AX, 6	SHRW AX, 6	SARW AX, 6
7	BTCLR saddr.7, \$addr20	BTCLR A.7, \$addr20	BT saddr.7, \$addr20	BT A.7, \$addr20	BF saddr.7, \$addr20	BF A.7, \$addr20		SHL C, 7	SHL B, 7	SHL A, 7	SHR A, 7	SAR A, 7	SHLW BC, 7	SHLW AX, 7	SHRW AX, 7	SARW AX, 7
8	BTCLR sfr.0, \$addr20	[HL].0, \$addr20	BT sfr.0, \$addr20	[HL].0, \$addr20	BF sfr.0, \$addr20	BF [HL].0, \$addr20							SHLW BC, 8	SHLW AX, 8	SHRW AX, 8	SARW AX, 8
9	BTCLR sfr.1, \$addr20	[HL].1, \$addr20	BT sfr.1, \$addr20	[HL].1, \$addr20	BF sfr.1, \$addr20	BF [HL].1, \$addr20							SHLW BC, 9	SHLW AX, 9	SHRW AX, 9	SARW AX, 9
a	BTCLR sfr.2, \$addr20	[HL].2, \$addr20	BT sfr.2, \$addr20	[HL].2, \$addr20	BF sfr.2, \$addr20	BF [HL].2, \$addr20							SHLW BC, 10	SHLW AX, 10	SHRW AX, 10	SARW AX, 10
b	BTCLR sfr.3, \$addr20	[HL].3, \$addr20	BT sfr.3, \$addr20	[HL].3, \$addr20	BF sfr.3, \$addr20	BF [HL].3, \$addr20							SHLW BC, 11	SHLW AX, 11	SHRW AX, 11	SARW AX, 11
c	BTCLR sfr.4, \$addr20	[HL].4, \$addr20	BT sfr.4, \$addr20	[HL].4, \$addr20	BF sfr.4, \$addr20	BF [HL].4, \$addr20							SHLW BC, 12	SHLW AX, 12	SHRW AX, 12	SARW AX, 12
d	BTCLR sfr.5, \$addr20	[HL].5, \$addr20	BT sfr.5, \$addr20	[HL].5, \$addr20	BF sfr.5, \$addr20	BF [HL].5, \$addr20							SHLW BC, 13	SHLW AX, 13	SHRW AX, 13	SARW AX, 13
e	BTCLR sfr.6, \$addr20	[HL].6, \$addr20	BT sfr.6, \$addr20	[HL].6, \$addr20	BF sfr.6, \$addr20	BF [HL].6, \$addr20							SHLW BC, 14	SHLW AX, 14	SHRW AX, 14	SARW AX, 14
f	BTCLR sfr.7, \$addr20	[HL].7, \$addr20	BT sfr.7, \$addr20	[HL].7, \$addr20	BF sfr.7, \$addr20	BF [HL].7, \$addr20							SHLW BC, 15	SHLW AX, 15	SHRW AX, 15	SARW AX, 15

Section 6 Explanation of Instructions

This section explains the instructions of RL78 microcontrollers.

Description Example



[Instruction format] MOV dst, src: Indicates the basic description format of the instruction.

[Operation] dst ← src: Indicates instruction operation using symbols.

[Operand] Indicates operands that can be specified by this instruction. See **5.2 Description in “Operation” Column** for the description of each operand symbol.

Mnemonic	Operand (dst, src)	Mnemonic	Operand (dst, src)
MOV	r, #byte	MOV	A, saddr
	≈PSW, #byte		≈saddr, A
	A, PSW		A, [HL+byte]
	≈PSW, A		≈[HL+byte], A

[Flag] Indicates flag behavior following execution of an instruction; see the legend for the symbols used.

Z	AC	CY

Legend

Symbol	Flag Behavior
Blank	Unchanged
0	Cleared to 0
1	Set to 1
×	Set or cleared according to the result
R	Previously saved value is restored

[Description] Describes the instruction operation in details.

- The content of the source operand (src) specified by the second operand is transferred to the destination operand (dst) specified by the first operand.

[Description example]

MOV A, #4DH; 4DH is transferred to the A register.

6.1 8-bit Data Transfer Instructions

The following instructions are used to transfer 8-bit data.

- MOV
- XCH
- ONEB
- CLRB
- MOVS

MOVMove
Byte Data Transfer

[Instruction format] MOV dst, src

[Operation] dst ← src

[Operand]

Mnemonic	Operand (dst, src)
MOV	r, #byte
	PSW, #byte
	CS, #byte
	ES, #byte
	!addr16, #byte
	ES:!addr16, #byte
	saddr, #byte
	sfr, #byte
	[DE+byte], #byte
	ES:[DE+byte], #byte
	[HL+byte], #byte
	ES:[HL+byte], #byte
	[SP+byte], #byte
	word[B], #byte
	ES:word[B], #byte
	word[C], #byte
	ES:word[C], #byte
	word[BC], #byte
	ES:word[BC], #byte
	A, r ^{Note}
	r, A ^{Note}
	A, PSW
	PSW, A
	CS, A
	A, CS
	ES, A
	A, ES
	A, !addr16
	A, ES:!addr16
	!addr16, A
	ES:!addr16, A

Mnemonic	Operand (dst, src)
MOV	A, saddr
	saddr, A
	A, sfr
	sfr, A
	A, [DE]
	[DE], A
	A, ES:[DE]
	ES:[DE], A
	A, [HL]
	[HL], A
	A, ES:[HL]
	ES:[HL], A
	A, [DE+byte]
	[DE+byte], A
	A, ES:[DE+byte]
	ES:[DE+byte], A
	A, [HL+byte]
	[HL+byte], A
	A, ES:[HL+byte]
	ES:[HL+byte], A
	A, [SP+byte]
	[SP+byte], A
	A, word[B]
	word[B], A
	A, ES:word[B]
	ES:word[B], A
	A, word[C]
	word[C], A
	A, ES:word[C]
	ES:word[C], A
	A, word[BC]

Mnemonic	Operand (dst, src)
MOV	word[BC], A
	A, ES:word[BC]
	ES:word[BC], A
	A, [HL+B]
	[HL+B], A
	A, ES:[HL+B]
	ES:[HL+B], A
	A, [HL+C]
	[HL+C], A
	A, ES:[HL+C]
	ES:[HL+C], A
	X, !addr16
	X, ES:!addr16
	X, saddr
	B, !addr16
	B, ES:!addr16
	B, saddr
	C, !addr16
	C, ES:!addr16
	C, saddr
	ES, saddr

Note Specify a register other than A.

[Flag]

PSW, #byte and PSW,
A operands

Z	AC	CY
x	x	x

All other operand combinations

Z	AC	CY

[Description]

- The content of the source operand (src) specified by the second operand is transferred to the destination operand (dst) specified by the first operand.
- No interrupt requests are acknowledged until the instruction following either MOV PSW, #byte or MOV PSW, A has been executed.

[Description example]

MOV A, #4DH; 4DH is transferred to the A register.

XCHExchange
Byte Data Transfer

[Instruction format] XCH dst, src

[Operation] dst $\longleftrightarrow$ src

[Operand]

Mnemonic	Operand (dst, src)
XCH	A, r ^{Note}
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, sfr
	A, [DE]
	A, ES:[DE]
	A, [HL]
	A, ES:[HL]

Mnemonic	Operand (dst, src)
XCH	A, [DE+byte]
	A, ES:[DE+byte]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY

[Description]

- The first and second operand contents are exchanged.

[Description example]

XCH A, FFEBCH; The content of the A register and that at address FFEBCH are exchanged.

ONEBOne byte
Byte Data 01H Set

[Instruction format] ONEB dst

[Operation] dst ← 01H

[Operand]

Mnemonic	Operand (dst)
ONEB	A
	X
	B
	C
	!addr16
	ES:!addr16
	saddr

[Flag]

Z	AC	CY

[Description]

- 01H is transferred to the destination operand (dst) specified by the first operand.

[Description example]

ONEB A; Transfers 01H to the A register.

CLRB

Clear byte
Byte Data Clear

[Instruction format] CLRB dst

[Operation] dst ← 00H

[Operand]

Mnemonic	Operand (dst)
CLRB	A
	X
	B
	C
	!addr16
	ES:!addr16
	saddr

[Flag]

Z	AC	CY

[Description]

- 00H is transferred to the destination operand (dst) specified by the first operand.

[Description example]

CLRB A; Transfers 00H to the A register.

MOVS

Move and change PSW
Byte Data Transfer and PSW Change

[Instruction format] MOVS dst, src

[Operation] dst ← src

[Operand]

Mnemonic	Operand (dst, src)
MOVS	[HL+byte], X
	ES:[HL+byte], X

[Flag]

Z	AC	CY
x		x

[Description]

- The content of the source operand specified by the second operand is transferred to the destination operand (dst) specified by the first operand.
- If the src value is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the register A value is 0 or the src value is 0, the CY flag is set to 1; otherwise, it is cleared to 0.

[Description example]

MOVS [HL+2H], X; When HL is FE00H, X is 55H, and A is 0H, the value of X (55H) is stored at address FE02H. The Z flag is 0, and the CY flag is 1 because the value of the A register is 0.

6.2 16-bit Data Transfer Instructions

The following instructions are used to transfer 16-bit data.

- MOVW
- XCHW
- ONEW
- CLRW

MOVWMove Word
Word Data Transfer

[Instruction format] MOVW dst, src

[Operation] dst ← src

[Operand]

Mnemonic	Operand (dst, src)
MOVW	rp, #word
	saddrp, #word
	sfrp, #word
	AX, rp ^{Note}
	rp, AX ^{Note}
	AX, !addr16
	!addr16, AX
	AX, ES:!addr16
	ES:!addr16, AX
	AX, saddrp
	saddrp, AX
	AX, sfrp
	sfrp, AX
	AX, [DE]
	[DE], AX
	AX, ES:[DE]
	ES:[DE], AX
	AX, [HL]
	[HL], AX
	AX, ES:[HL]
	ES:[HL], AX
	AX, [DE+byte]
	[DE+byte], AX
	AX, ES:[DE+byte]
	ES:[DE+byte], AX
	AX, [HL+byte]

Mnemonic	Operand (dst, src)
MOVW	[HL+byte], AX
	AX, ES:[HL+byte]
	ES:[HL+byte], AX
	AX, [SP+byte]
	[SP+byte], AX
	AX, word[B]
	word[B], AX
	AX, ES:word[B]
	ES:word[B], AX
	AX, word[C]
	word[C], AX
	AX, ES:word[C]
	ES:word[C], AX
	AX, word[BC]
	word[BC], AX
	AX, ES:word[BC]
	ES:word[BC], AX
	BC, !addr16
	BC, ES:!addr16
	DE, !addr16
	DE, ES:!addr16
	HL, !addr16
	HL, ES:!addr16
	BC, saddrp
	DE, saddrp
	HL, saddrp

Note Only specify BC, DE, or HL.

[Flag]

Z	AC	CY

[Description]

- The content of the source operand (src) specified by the second operand are transferred to the destination operand (dst) specified by the first operand.

[Description example]

MOVW AX, HL; The content of the HL register is transferred to the AX register.

[Caution]

Only an even address can be specified.

XCHWExchange Word
Word Data Exchange

[Instruction format] XCHW dst, src

[Operation] dst $\longleftrightarrow$ src

[Operand]

Mnemonic	Operand (dst, src)
XCHW	AX, rp ^{Note}

Note Only specify BC, DE, or HL.

[Flag]

Z	AC	CY

[Description]

- The first and second operand contents are exchanged.

[Description example]

XCHW AX, BC; The contents of the AX and BC registers are exchanged.

ONEW

One Word

Word Data 0001 Set

[Instruction format] ONEW dst

[Operation] dst ← 0001H

[Operand]

Mnemonic	Operand (dst)
ONEW	AX
	BC

[Flag]

Z	AC	CY

[Description]

- 0001H is transferred to the destination operand (dst) specified by the first operand.

[Description example]

ONEW AX; 0001H is transferred to the AX register.

CLRW

Clear Word
Word Data Clear

[Instruction format] CLRW dst

[Operation] dst ← 0000H

[Operand]

Mnemonic	Operand (dst)
CLRW	AX
	BC

[Flag]

Z	AC	CY

[Description]

- 0000H is transferred to the destination operand (dst) specified by the first operand.

[Description example]

CLRW AX; 0000H is transferred to the AX register.

6.3 8-bit Operation Instructions

The following instructions are used for arithmetic and logical operations on 8-bit values.

- ADD
- ADDC
- SUB
- SUBC
- AND
- OR
- XOR
- CMP
- CMP0
- CMPS

ADDAdd
Byte Data Addition

[Instruction format] ADD dst, src

[Operation] dst, CY ← dst + src

[Operand]

Mnemonic	Operand (dst, src)
ADD	A, #byte
	saddr, #byte
	A, r ^{Note}
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
ADD	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
x	x	x

[Description]

- The destination operand (dst) specified by the first operand is added to the source operand (src) specified by the second operand, and the result is stored in the destination operand (dst).
- If the addition result shows that dst is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the addition generates a carry from bit 7, the CY flag is set to 1; otherwise, it is cleared to 0.
- If the addition generates a carry from bit 3 to bit 4, the AC flag is set to 1; otherwise, it is cleared to 0.

[Description example]

ADD CR10, #56H; 56H is added to the CR10 register and the result is stored in the CR10 register.

ADDC

Add with Carry
Addition of Byte Data with Carry

[Instruction format] ADDC dst, src

[Operation] $\text{dst, CY} \leftarrow \text{dst} + \text{src} + \text{CY}$

[Operand]

Mnemonic	Operand (dst, src)
ADDC	A, #byte
	saddr, #byte
	A, r ^{Note}
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
ADDC	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
x	x	x

[Description]

- The destination operand (dst) specified by the first operand, the source operand (src) specified by the second operand, and the value of the CY flag are added; the result is stored in the destination operand (dst). The value of the CY flag is added at the least significant bit. This instruction is mainly used for multibyte addition.
- If the addition result shows that dst is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the addition generates a carry from bit 7, the CY flag is set to 1; otherwise, it is cleared to 0.
- If the addition generates a carry from bit 3 to bit 4, the AC flag is set to 1; otherwise, it is cleared to 0.

[Description example]

ADDC A, [HL+B]; The content of the A register, that at the address indicated by the HL and B registers, and the value of the CY flag are added, and the result is stored in the A register.

SUB

Subtract
Byte Data Subtraction

[Instruction format] SUB dst, src

[Operation] dst, CY ← dst - src

[Operand]

Mnemonic	Operand (dst, src)
SUB	A, #byte
	saddr, #byte
	A, r ^{Note}
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
SUB	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
x	x	x

[Description]

- The source operand (src) specified by the second operand is subtracted from the destination operand (dst) specified by the first operand, and the result is stored in the destination operand (dst). The destination operand (dst) can be cleared to 0 by specifying the same operand for both the source operand (src) and the destination operand (dst).
- If the subtraction shows that dst is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 7, the CY flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 4 to bit 3, the AC flag is set to 1; otherwise, it is cleared to 0.

[Description example]

SUB D, A; The content of the A register is subtracted from that of the D register, and the result is stored in the D register.

SUBC

Subtract with Carry
Subtraction of Byte Data with Carry

[Instruction format] SUBC dst, src

[Operation] $\text{dst, CY} \leftarrow \text{dst} - \text{src} - \text{CY}$

[Operand]

Mnemonic	Operand (dst, src)
SUBC	A, #byte
	saddr, #byte
	A, r ^{Note}
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
SUBC	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
x	x	x

[Description]

- The source operand (src) specified by the second operand and the value of the CY flag are subtracted from the destination operand (dst) specified by the first operand, and the result is stored in the destination operand (dst). The value of the CY flag is subtracted from the least significant bit. This instruction is mainly used for multibyte subtraction.
- If the subtraction shows that dst is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 7, the CY flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 4 to bit 3, the AC flag is set to 1; otherwise, it is cleared to 0.

[Description example]

SUBC A, [HL]; The content at the address indicated by the HL register and the value of the CY flag are subtracted from the content of the A register, and the result is stored in the A register.

AND

And
Logical Product of Byte Data

[Instruction format] AND dst, src

[Operation] $\text{dst} \leftarrow \text{dst} \wedge \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
AND	A, #byte
	saddr, #byte
	A, r ^{Note}
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
AND	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
×		

[Description]

- The destination operand (dst) specified by the first operand is bitwise ANDed with the source operand (src) specified by the second operand, and the result is stored in the destination operand (dst).
- If all bits in the result are 0, the Z flag is set to 1; otherwise, it is cleared to 0.

[Description example]

AND FFEBAH, #11011100B; The content at address FFEBAH is bitwise ANDed with 11011100B, and the result is stored at address FFEBAH.

OROr
Logical Sum of Byte Data

[Instruction format] OR dst, src

[Operation] $\text{dst} \leftarrow \text{dst} \vee \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
OR	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
OR	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
×		

[Description]

- The destination operand (dst) specified by the first operand is bitwise ORed with the source operand (src) specified by the second operand, and the result is stored in the destination operand (dst).
- If all bits in the result are 0, the Z flag is set to 1; otherwise, it is cleared to 0.

[Description example]

OR A, FFE98H; The content of the A register is bitwise ORed with the content at address FFE98H, and the result is stored in the A register.

XORExclusive Or
Exclusive Logical Sum of Byte Data

[Instruction format] XOR dst, src

[Operation] $\text{dst} \leftarrow \text{dst} \vee \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
XOR	A, #byte
	saddr, #byte
	A, r ^{Note}
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr
	A, [HL]

Mnemonic	Operand (dst, src)
XOR	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
×		

[Description]

- The destination operand (dst) specified by the first operand is bitwise XORed with the source operand (src) specified by the second operand, and the result is stored in the destination operand (dst). All bits of the destination operand (dst) can be logically negated by specifying #0FFH for the source operand (src).
- If all bits in the result are 0, the Z flag is set to 1; otherwise, it is cleared to 0.

[Description example]

XOR A, L; The content of the A register is bitwise XORed with that of the L register, and the result is stored in the A register.

CMPCompare
Byte Data Comparison

[Instruction format] CMP dst, src

[Operation] dst - src

[Operand]

Mnemonic	Operand (dst, src)
CMP	A, #byte
	!addr16, #byte
	ES:!addr16, #byte
	saddr, #byte
	A, r Note
	r, A
	A, !addr16
	A, ES:!addr16
	A, saddr

Mnemonic	Operand (dst, src)
CMP	A, [HL]
	A, ES:[HL]
	A, [HL+byte]
	A, ES:[HL+byte]
	A, [HL+B]
	A, ES:[HL+B]
	A, [HL+C]
	A, ES:[HL+C]

Note Specify a register other than A.

[Flag]

Z	AC	CY
x	x	x

[Description]

- The source operand (src) specified by the second operand is subtracted from the destination operand (dst) specified by the first operand. The result of the subtraction is discarded, and only the Z, AC, and CY flags are updated.
- If the subtraction result is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 7, the CY flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 4 to bit 3, the AC flag is set to 1; otherwise, it is cleared to 0.

[Description example]

CMP FFE38H, #38H; The content at address FFE38H and the immediate data are compared by subtracting 38H from the content; only the flags are updated.

CMP0

Compare 00H
Byte Data Zero Comparison

[Instruction format] CMP0 dst

[Operation] dst - 00H

[Operand]

Mnemonic	Operand (dst)
CMP0	A
	X
	B
	C
	!addr16
	ES:!addr16
	saddr

[Flag]

Z	AC	CY
×	0	0

[Description]

- 00H is subtracted from the destination operand (dst) specified by the first operand.
- The result of the subtraction is discarded, and only the Z, AC, and CY flags are updated.
- If the dst value is 00H, the Z flag is set to 1; otherwise, it is cleared to 0.
- The AC and CY flags are always cleared to 0.

[Description example]

CMP0 A; The Z flag is set if the A register value is 0.

CMPSCompare
Byte Data Comparison

[Instruction format] CMPS dst, src

[Operation] dst - src

[Operand]

Mnemonic	Operand (dst, src)
CMPS	X, [HL+byte]
	X, ES:[HL+byte]

[Flag]

Z	AC	CY
x	x	x

[Description]

- The source operand (src) specified by the second operand is subtracted from the destination operand (dst) specified by the first operand. The result of the subtraction is discarded, and only the Z, AC, and CY flags are updated.
- If the subtraction result is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- When the calculation result is not 0, or the value of either register A or dst is 0, the CY flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 4 to bit 3, the AC flag is set to 1; otherwise, it is cleared to 0.

[Description example]

CMPS X, [HL+F0H]; When HL = FD12H

- The value of X is compared with the content at address FFE02H, and the Z flag is set if the two values match.
- The value of X is compared with the content at address FFE02H, and the CY flag is set if the two values do not match.
- The CY flag is set if the value of the A register is 0.
- The CY flag is set if the value of the X register is 0.
- The AC flag is set by a borrow from bit 4 to bit 3, as with the CMP instruction.

6.4 16-bit Operation Instructions

The following instructions are used to add, subtract, or compare 16-bit values.

- ADDW
- SUBW
- CMPW

ADDWAdd Word
Word Data Addition

[Instruction format] ADDW dst, src

[Operation] dst, CY ← dst + src

[Operand]

Mnemonic	Operand (dst, src)
ADDW	AX, #word
	AX, AX
	AX, BC
	AX, DE
	AX, HL
	AX, !addr16
	AX, ES:!addr16
	AX, saddrp
	AX, [HL+byte]
	AX, ES:[HL+byte]

[Flag]

Z	AC	CY
x	x	x

[Description]

- The destination operand (dst) specified by the first operand is added to the source operand (src) specified by the second operand and the result is stored in the destination operand (dst).
- If the addition result shows that dst is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the addition generates a carry from bit 15, the CY flag is set to 1; otherwise, it is cleared to 0.
- The value of the AC flag after the addition is undefined.

[Description example]

ADDW AX, #ABCDH; ABCDH is added to the content of the AX register, and the result is stored in the AX register.

SUBW

Subtract Word
Word Data Subtraction

[Instruction format] SUBW dst, src

[Operation] dst, CY ← dst - src

[Operand]

Mnemonic	Operand (dst, src)
SUBW	AX, #word
	AX, BC
	AX, DE
	AX, HL
	AX, !addr16
	AX, ES:!addr16
	AX, saddrp
	AX, [HL+byte]
	AX, ES:[HL+byte]

[Flag]

Z	AC	CY
x	x	x

[Description]

- The source operand (src) specified by the second operand is subtracted from the destination operand (dst) specified by the first operand and the result is stored in the destination operand (dst).
- If the subtraction shows that dst is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 15, the CY flag is set to 1; otherwise, it is cleared to 0.
- The value of the AC flag after the subtraction is undefined.

[Description example]

SUBW AX, #ABCDH; ABCDH is subtracted from the content of the AX register, and the result is stored in the AX register.

CMPWCompare Word
Word Data Comparison

[Instruction format] CMPW dst, src

[Operation] dst - src

[Operand]

Mnemonic	Operand (dst, src)
CMPW	AX, #word
	AX, BC
	AX, DE
	AX, HL
	AX, !addr16
	AX, ES:!addr16
	AX, saddrp
	AX, [HL+byte]
	AX, ES:[HL+byte]

[Flag]

Z	AC	CY
x	x	x

[Description]

- The destination operand (dst) specified by the first operand is compared with the source operand (src) specified by the second operand by subtraction; the result of the subtraction is discarded, and only the Z, AC, and CY flags are updated.
- If the subtraction result is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the subtraction generates a borrow from bit 15, the CY flag is set to 1; otherwise, it is cleared to 0.
- The value of the AC flag after the subtraction is undefined.

[Description example]

CMPW AX, #ABCDH; The content of the AX register and the immediate data are compared by subtracting ABCDH from the content; only the flags are updated.

6.5 Multiply, Divide, and Multiply-and-accumulate Instructions

The following instructions are used to multiply, divide, or multiply and accumulate values.

- MULU
- MULHU
- MULH
- DIVHU
- DIVWU
- MACHU
- MACH

Caution The following multiply, divide, and multiply-and-accumulate instructions are extended instructions and only included in the instruction set of the RL78-S3 core.

- **MULHU (16-bit multiplication unsigned)**
- **MULH (16-bit multiplication signed)**
- **DIVHU (16-bit division unsigned)**
- **DIVWU (32-bit division unsigned)**
- **MACHU (16-bit multiplication and accumulation unsigned (16 bits × 16 bits) + 32 bits)**
- **MACH (16-bit multiplication and accumulation signed (16 bits × 16 bits) + 32 bits)**

MULU

Multiply Unsigned
Unsigned Multiplication of Data

[Instruction format] MULU src

[Operation] $AX \leftarrow A \times \text{src}$

[Operand]

Mnemonic	Operand (src)
MULU	X

[Flag]

Z	AC	CY

[Description]

- The content of the A register is multiplied with the source operand (src) as unsigned data, and the result is stored in the AX register.

[Description example]

MULU X; The content of the A register is multiplied with the data of the source operand (src), and the result is stored in the AX register.

MULHU

Multiply Unsigned
Unsigned Multiplication of Data

[Instruction format] MULHU

[Operation] $BCAX \leftarrow AX \times BC$

[Operand]

Mnemonic	Operand (src)
MULHU	

[Flag]

Z	AC	CY

[Description]

- The contents of the AX and BC registers are multiplied as unsigned data, and the 16 higher-order bits of the result are stored in the BC register and the 16 lower-order bits are stored in the AX register.

[Description example]

MOVW AX, #0C000H

MOVW BC, #1000H

MULHU

MOVW !addr16, AX

MOVW AX, BC

MOVW !addr16, AX; C000H and 1000H are multiplied, and the result, C000000H, is stored in memory at the address indicated by !addr16.

MULH

Multiply Signed
Signed Multiplication of Data

[Instruction format] MULH

[Operation] $BCAX \leftarrow AX \times BC$

[Operand]

Mnemonic	Operand (src)
MULH	

[Flag]

Z	AC	CY

[Description]

- The contents of the AX and BC registers are multiplied as signed data, and the 16 higher-order bits of the result are stored in the BC register and the 16 lower-order bits are stored in the AX register.

[Description example]

MOVW AX, #0C000H

MOVW BC, #1000H

MULH

MOVW !addr16, AX

MOVW AX, BC

MOVW !addr16, AX; C000H and 1000H are multiplied, and the result, FC000000H, is stored in memory at the address indicated by !addr16.

DIVHU16-bit Divide Unsigned
Unsigned Division of Data

[Instruction format] DIVHU

[Operation] AX (quotient), DE (remainder) $\leftarrow$ AX $\div$ DE

[Operand]

Mnemonic	Operand (src)
DIVHU	

[Flag]

Z	AC	CY

[Description]

- The content of the AX register is divided by that of the DE register, and the quotient and remainder are stored in the AX and DE registers, respectively. The contents of the AX and DE registers are treated as unsigned data in the division. However, if the content of the DE register is 0, the content of the AX register is stored in the DE register, and the AX register is set to 0FFFFH.

[Description example]

MOVW AX, #8081H

MOVW DE, #0002H

DIVHU

MOVW !addr16, AX

MOVW AX, DE

MOVW !addr16, AX; 8081H is divided by 0002H, and the quotient (4040H) in the AX register and the remainder (0001H) in the DE register are stored in memory at the address indicated by !addr16.

<R> **Caution** Disable interrupts when executing the DIVHU or DIVWU instruction in an interrupt servicing routine. Alternatively, unless they are executed in the RAM area, note that execution of a DIVHU or DIVWU instruction is possible even with interrupts enabled as long as a NOP instruction is added immediately after the DIVHU or DIVWU instruction in the assembly language source code. For C language source code, depending on the C compiler, the NOP instruction may automatically be added immediately after the DIVHU or DIVWU instruction is output during the build process. Confirm the specifications of the C compiler you will be using.

DIVWU32-bit Divide Unsigned
Unsigned Division of Data

[Instruction format] DIVWU

[Operation] BCAX (quotient), HLDE (remainder) ← BCAX ÷ HLDE

[Operand]

Mnemonic	Operand (src)
DIVWU	

[Flag]

Z	AC	CY

[Description]

- The content of the BCAX register is divided by that of the HLDE register, and the quotient and remainder are stored in the BCAX and HLDE registers, respectively. The contents of the BCAX and HLDE registers are treated as unsigned data in the division. However, if the content of the HLDE register is 0, the content of the BCAX register is stored in the HLDE register, and the BCAX register is set to 0FFFFFFFH.

[Description example]

```

MOVW AX, #8081H
MOVW BC, #8080H
MOVW DE, #0002H
MOVW HL, #0000H
DIVWU
MOVW !addr16, AX
MOVW AX, BC
MOVW !addr16, AX
MOVW AX, DE
MOVW !addr16, AX
MOVW AX, HL

```

MOVW !addr16, AX; 80808081H is divided by 00000002H, and the quotient (40404040H) in the BCAX register and the remainder (00000001H) in the HLDE register are stored in memory at the address indicated by !addr16.

<R> **Caution** Disable interrupts when executing the DIVHU or DIVWU instruction in an interrupt servicing routine. Alternatively, unless they are executed in the RAM area, note that execution of a DIVHU or DIVWU instruction is possible even with interrupts enabled as long as a NOP instruction is added immediately after the DIVHU or DIVWU instruction in the assembly language source code. For C language source code, depending on the C compiler, the NOP instruction may automatically be added immediately after the DIVHU or DIVWU instruction is output during the build process. Confirm the specifications of the C compiler you will be using.

MACHU

Multiply and Accumulate Unsigned
Unsigned Multiplication and Accumulation of Data

[Instruction format] MACHU

[Operation] $MACR \leftarrow MACR + AX \times BC$

[Operand]

Mnemonic	Operand (src)
MACHU	

[Flag]

Z	AC	CY
	x	x

[Description]

- <R>
- The contents of the AX and BC registers are multiplied as unsigned data, and the result is added to the content of the MACR register and stored back.
 - If overflow occurs during accumulation, the CY flag is set to 1; otherwise, it is cleared to 0.
 - The AC flag becomes 0.
 - Set an initial value in the MACR register before multiplication and accumulation. Since only one MACR register is present, save its content before using it for the next operation if multiple results of multiplication and accumulation are required.

[Description example]

```
MOVW AX, #00000H
MOVW !0FFF2H, AX
MOVW !0FFF0H, AX
MOVW AX, #0C000H
MOVW BC, #01000H
MACHU
MOVW AX, !0FFF2H
MOVW !addr16, AX
MOVW AX, !0FFF0H
```

MOVW !addr16, AX; The contents of the AX and BC registers are multiplied, and the result is added to the content of the MACR register. The addition result is stored in the MACR register.

MACH

Multiply and Accumulate Signed
Signed Multiplication and Accumulation of Data

[Instruction format] MACH

[Operation] $MACR \leftarrow MACR + AX \times BC$

[Operand]

Mnemonic	Operand (src)
MACH	

[Flag]

Z	AC	CY
	x	x

[Description]

<R>

- The contents of the AX and BC registers are multiplied as signed data, and the result is added to the content of the MACR register and stored back.
- If overflow occurs during accumulation, the CY flag is set to 1; if not, it is cleared to 0. Overflow occurs when the sum of a positive accumulated value and a positive product exceeds 7FFFFFFFH, or when the sum of a negative accumulated value and a negative product exceeds 80000000H.
- The AC flag is cleared to 0 if the content of the MACR register is positive, and set to 1 if it is negative.
- Set an initial value in the MACR register before multiplication and accumulation. Since only one MACR register is present, save its content before using it for the next operation if multiple results of multiplication and accumulation are required.

[Description example]

```
MOVW AX, #00000H
MOVW !0FFF0H, AX
MOVW AX, #08000H
MOVW !0FFF2H, AX
MOVW AX, #00001H
MOVW !0FFF0H, AX
MOVW AX, #07FFFH
MOVW BC, #0FFFFH
MACH
MOVW AX, !0FFF2H
MOVW !addr16, AX
MOVW AX, !0FFF0H
```

MOVW !addr16, AX; The contents of the AX and BC registers are multiplied, and the result is added to the content of the MACR register. The addition result is stored in the MACR register.

6.6 Increment and Decrement Instructions

The following instructions are used to increment or decrement a value.

- INC
- DEC
- INCW
- DECW

INCIncrement
Byte Data Increment

[Instruction format] INC dst

[Operation] $\text{dst} \leftarrow \text{dst} + 1$

[Operand]

Mnemonic	Operand (dst)
INC	r
	!addr16
	ES:!addr16
	saddr
	[HL+byte]
	ES:[HL+byte]

[Flag]

Z	AC	CY
x	x	

[Description]

- The content of the destination operand (dst) is incremented by 1.
- If the increment result is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the increment generates a carry from bit 3 to bit 4, the AC flag is set to 1; otherwise, it is cleared to 0.
- Since this instruction is frequently used to increment a counter for repeated operations or an offset register for indexed addressing, the CY flag is not updated so that its value is retained for multibyte operations.

[Description example]

INC B; The B register is incremented.

DECDecrement
Byte Data Decrement

[Instruction format] DEC dst

[Operation] $\text{dst} \leftarrow \text{dst} - 1$

[Operand]

Mnemonic	Operand (dst)
DEC	r
	!addr16
	ES:!addr16
	saddr
	[HL+byte]
	ES:[HL+byte]

[Flag]

Z	AC	CY
x	x	

[Description]

- The content of the destination operand (dst) is decremented by 1.
- If the decrement result is 0, the Z flag is set to 1; otherwise, it is cleared to 0.
- If the decrement generates a carry from bit 3 to bit 4, the AC flag is set to 1; otherwise, it is cleared to 0.
- Since this instruction is frequently used for a counter in repeated operations, the CY flag is not updated so that its value is retained for multibyte operations.
- When the B or C register or saddr is specified as dst, use the DBNZ instruction to leave the AC and CY flags unchanged.

[Description example]

DEC FFE92H; The content at address FFE92H is decremented.

INCW

Increment Word
Word Data Increment

[Instruction format] INCW dst

[Operation] $\text{dst} \leftarrow \text{dst} + 1$

[Operand]

Mnemonic	Operand (dst)
INCW	rp
	!addr16
	ES:!addr16
	saddrp
	[HL+byte]
	ES:[HL+byte]

[Flag]

Z	AC	CY

[Description]

- The content of the destination operand (dst) is incremented by 1.
- Since this instruction is frequently used to increment a register (pointer) used for addressing, the Z, AC, and CY flags are not updated so that flag values set by other instructions are not lost.

[Description example]

INCW HL; The HL register is incremented.

DECWDecrement Word
Word Data Decrement

[Instruction format] DECW dst

[Operation] $\text{dst} \leftarrow \text{dst} - 1$

[Operand]

Mnemonic	Operand (dst)
DECW	rp
	!addr16
	ES:!addr16
	saddrp
	[HL+byte]
	ES:[HL+byte]

[Flag]

Z	AC	CY

[Description]

- The content of the destination operand (dst) is decremented by 1.
- Since this instruction is frequently used to increment a register (pointer) used for addressing, the Z, AC, and CY flags are not updated so that flag values set by other instructions are not lost.

[Description example]

DECW DE; The DE register is decremented.

6.7 Shift Instructions

The following instructions are used to shift bits.

- SHR
- SHRW
- SHL
- SHLW
- SAR
- SARW

SHRShift Right
Logical Shift to the Right

[Instruction format] SHR dst, cnt

[Operation] $(CY \leftarrow dst_0, dst_{m-1} \leftarrow dst_m, dst_7 \leftarrow 0) \times cnt$

[Operand]

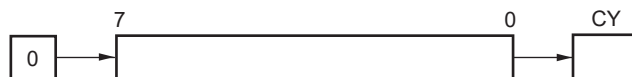
Mnemonic	Operand (dst, cnt)
SHR	A, cnt

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) specified by the first operand is shifted to the right the number of times specified by cnt.
- “0” is entered to the MSB (bit 7) and the value shifted last from bit 0 is entered to CY.
- cnt can be specified as any value from 1 to 7.



[Description example]

SHR A, 3; When the value of the A register is F5H, A = 1EH and CY = 1.

A = 1111_0101B CY = 0

A = 0111_1010B CY = 1 1 time

A = 0011_1101B CY = 0 2 times

A = 0001_1110B CY = 1 3 times

SHRWShift Right Word
Logical Shift to the Right

[Instruction format] SHRW dst, cnt

[Operation] $(CY \leftarrow dst_0, dst_{m-1} \leftarrow dst_m, dst_{15} \leftarrow 0) \times cnt$

[Operand]

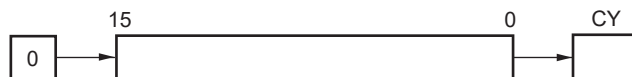
Mnemonic	Operand (dst, cnt)
SHRW	AX, cnt

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) specified by the first operand is shifted to the right the number of times specified by cnt.
- “0” is entered to the MSB (bit 15) and the value shifted last from bit 0 is entered to CY.
- cnt can be specified as any value from 1 to 15.



[Description example]

SHRW AX 3; When the value of the AX register is AAF5H, AX = 155EH and CY = 1.

AX = 1010_1010_1111_0101B	CY = 0
AX = 0101_0101_0111_1010B	CY = 1 1 time
AX = 0010_1010_1011_1101B	CY = 0 2 times
AX = 0001_0101_0101_1110B	CY = 1 3 times

SHLShift Left
Logical Shift to the Left

[Instruction format] SHL dst, cnt

[Operation] $(CY \leftarrow dst7, dstm \leftarrow dstm-1, dst0 \leftarrow 0) \times cnt$

[Operand]

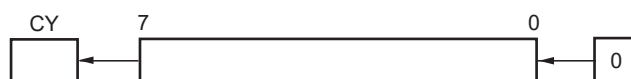
Mnemonic	Operand (dst, cnt)
SHL	A, cnt
	B, cnt
	C, cnt

[Flag]

Z	AC	CY
		x

[Description]

- The destination operand (dst) specified by the first operand is shifted to the left the number of times specified by cnt.
- "0" is entered to the LSB (bit 0) and the value shifted last from bit 7 is entered to CY.
- cnt can be specified as any value from 1 to 7.



[Description example]

SHL A, 3; When the value of the A register is 5DH, A = E8H and CY = 0.

CY = 0 A = 0101_1101B

CY = 0 A = 1011_1010B 1 time

CY = 1 A = 0111_0100B 2 times

CY = 0 A = 0110_1000B 3 times

SHLWShift Left Word
Logical Shift to the Left

[Instruction format] SHLW dst, cnt

[Operation] $(CY \leftarrow dst_{15}, dst_m \leftarrow dst_{m-1}, dst_0 \leftarrow 0) \times cnt$

[Operand]

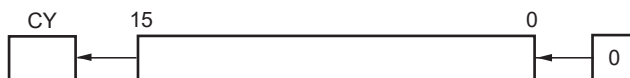
Mnemonic	Operand (dst, cnt)
SHLW	AX, cnt
	BC, cnt

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) specified by the first operand is shifted to the left the number of times specified by cnt.
- "0" is entered to the LSB (bit 0) and the value shifted last from bit 15 is entered to CY.
- cnt can be specified as any value from 1 to 15.



[Description example]

SHLW BC, 3; When the value of the BC register is C35DH, BC = 1AE8H and CY = 0.

CY = 0 BC = 1100_0011_0101_1101B

CY = 1 BC = 1000_0110_1011_1010B 1 time

CY = 1 BC = 0000_1101_0111_0100B 2 times

CY = 0 BC = 0001_1010_1110_1000B 3 times

SARShift Arithmetic Right
Arithmetic Shift to the Right

[Instruction format] SAR dst, cnt

[Operation] $(CY \leftarrow dst0, dstm-1 \leftarrow dstm, dst7 \leftarrow dst7) \times cnt$

[Operand]

<R>

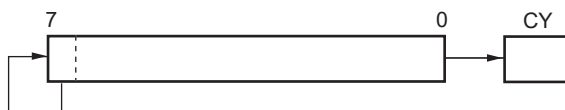
Mnemonic	Operand (dst, cnt)
SAR	A, cnt

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) specified by the first operand is shifted to the right the number of times specified by cnt.
- The same value is retained in the MSB (bit 7), and the value shifted last from bit 0 is entered to CY.
- cnt can be specified as any value from 1 to 7.



[Description example]

SAR A, 4; When the value of the A register is 8CH, A = F8H and CY = 1.

A = 1000_1100B CY = 0

A = 1100_0110B CY = 0 1 time

A = 1110_0011B CY = 0 2 times

A = 1111_0001B CY = 1 3 times

A = 1111_1000B CY = 1 4 times

SARWShift Arithmetic Right Word
Arithmetic Shift to the Right

[Instruction format] SARW dst, cnt

[Operation] $(CY \leftarrow dst0, dstm-1 \leftarrow dstm, dst15 \leftarrow dst15) \times cnt$

[Operand]

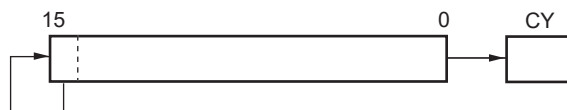
Mnemonic	Operand (dst, cnt)
SARW	AX, cnt

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) specified by the first operand is shifted to the right the number of times specified by cnt.
- The same value is retained in the MSB (bit 15), and the value shifted last from bit 0 is entered to CY.
- cnt can be specified as any value from 1 to 15.



[Description example]

SAR AX, 4; When the value of the AX register is A28CH, AX = FA28H and CY = 1.

AX = 1010_0010_1000_1100B CY = 0

AX = 1101_0001_0100_0110B CY = 0 1 time

AX = 1110_1000_1010_0011B CY = 0 2 times

AX = 1111_0100_0101_0001B CY = 1 3 times

AX = 1111_1010_0010_1000B CY = 1 4 times

6.8 Rotate Instructions

The following instructions are used to rotate bits.

- ROR
- ROL
- RORC
- ROLC
- ROLWC

ROR

Rotate Right
Byte Data Rotation to the Right

[Instruction format] ROR dst, cnt

[Operation] $(CY, dst7 \leftarrow dst0, dstm-1 \leftarrow dstm) \times \text{one time}$

[Operand]

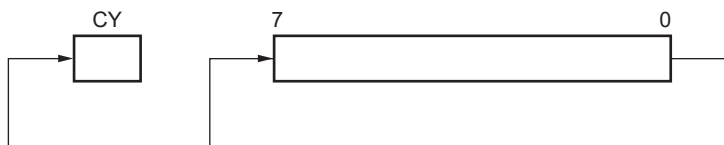
Mnemonic	Operand (dst, cnt)
ROR	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The content of the destination operand (dst) specified by the first operand is rotated to the right by one bit.
- The content of the LSB (bit 0) is simultaneously rotated to the MSB (bit 7) and transferred to the CY flag.



[Description example]

ROR A, 1; The content of the A register is rotated to the right by one bit.

ROL

Rotate Left
Byte Data Rotation to the Left

[Instruction format] ROL dst, cnt

[Operation] $(CY, dst_0 \leftarrow dst_7, dst_{m+1} \leftarrow dst_m) \times \text{one time}$

[Operand]

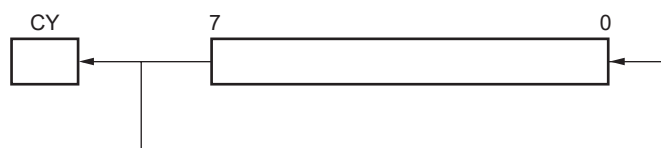
Mnemonic	Operand (dst, cnt)
ROL	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The content of the destination operand (dst) specified by the first operand is rotated to the right by one bit.
- The content of the MSB (bit 7) is simultaneously rotated to the LSB (bit 0) and transferred to the CY flag.



[Description example]

ROL A, 1; The content of the A register is rotated to the left by one bit.

RORC

Rotate Right with Carry
Byte Data Rotation to the Right with Carry

[Instruction format] RORC dst, cnt

[Operation] $(CY \leftarrow dst_0, dst_7 \leftarrow CY, dst_{m-1} \leftarrow dst_m) \times \text{one time}$

[Operand]

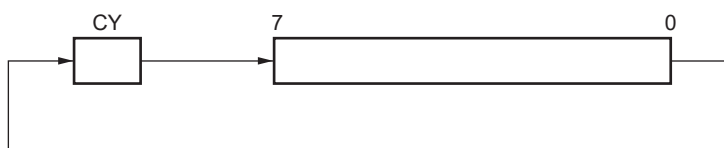
Mnemonic	Operand (dst, cnt)
RORC	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The content of the destination operand (dst) specified by the first operand and the CY flag are rotated to the right by one bit.



[Description example]

RORC A, 1; The content of the A register and the CY flag are rotated to the right by one bit.

ROLC

Rotate Left with Carry
Byte Data Rotation to the Left with Carry

[Instruction format] ROLC dst, cnt

[Operation] $(CY \leftarrow dst7, dst0 \leftarrow CY, dstm+1 \leftarrow dstm) \times \text{one time}$

[Operand]

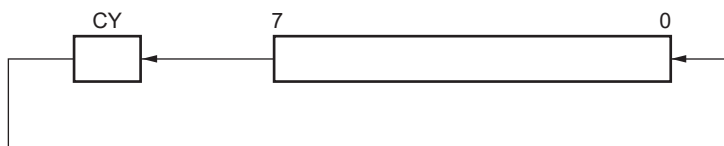
Mnemonic	Operand (dst, cnt)
ROLC	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The content of the destination operand (dst) specified by the first operand and the CY flag are rotated to the left by one bit.



[Description example]

ROLC A, 1; The content of the A register and the CY flag are rotated to the left by one bit.

ROLWC

Rotate Left word with Carry
Word Data Rotation to the Left with Carry

[Instruction format] ROLWC dst, cnt

[Operation] $(CY \leftarrow dst_{15}, dst_0 \leftarrow CY, dst_{m+1} \leftarrow dst_m) \times \text{one time}$

[Operand]

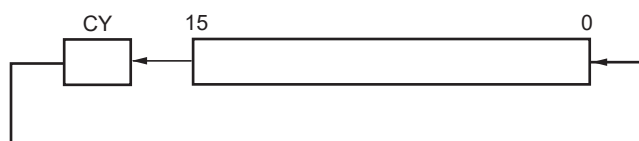
Mnemonic	Operand (dst, cnt)
ROLWC	AX, 1
	BC, 1

[Flag]

Z	AC	CY
		×

[Description]

- The content of the destination operand (dst) specified by the first operand and the CY flag are rotated to the left by one bit.



[Description example]

ROLWC BC, 1; The content of the BC register and the CY flag are rotated to the left by one bit.

6.9 Bit Manipulation Instructions

The following instructions are used to manipulate individual bits.

- MOV1
- AND1
- OR1
- XOR1
- SET1
- CLR1
- NOT1

MOV1Move Single Bit
1 Bit Data Transfer

[Instruction format] MOV1 dst, src

[Operation] $\text{dst} \leftarrow \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
MOV1	CY, A.bit
	A.bit, CY
	CY, PSW.bit
	PSW.bit, CY
	CY, saddr.bit
	saddr.bit, CY

Mnemonic	Operand (dst, src)
MOV1	CY, sfr.bit
	sfr.bit, CY
	CY, [HL].bit
	[HL].bit, CY
	CY, ES:[HL].bit
	ES:[HL].bit, CY

[Flag]

dst = CY

Z	AC	CY
		x

dst = PSW.bit

Z	AC	CY
x	x	

In all other cases

Z	AC	CY

[Description]

- The bit data of the source operand (src) specified by the second operand is transferred to the destination operand (dst) specified by the first operand.
- When CY or PSW.bit is specified as the destination operand (dst), only the corresponding flag changes.
- No interrupt requests are acknowledged until the instruction following MOV1 PSW.bit, CY has been executed.

[Description example]

MOV1 P3.4, CY; The value of the CY flag is transferred to bit 4 of port 3.

AND1

And Single Bit
1 Bit Data Logical Product

[Instruction format] AND1 dst, src

[Operation] $\text{dst} \leftarrow \text{dst} \wedge \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
AND1	CY, A.bit
	CY, PSW.bit
	CY, saddr.bit
	CY, sfr.bit
	CY, [HL].bit
	CY, ES:[HL].bit

[Flag]

Z	AC	CY
		x

[Description]

- The bit data of the destination operand (dst) specified by the first operand is ANDed with that of the source operand (src) specified by the second operand, and the result is stored in the destination operand.
- The operation result is stored in the CY flag because it is specified as dst.

[Description example]

AND1 CY, FFE7FH.3; The value of the CY flag is ANDed with bit 3 of FFE7FH, and the result is stored in the CY flag.

OR1

Or Single Bit
1 Bit Data Logical Sum

[Instruction format] OR1 dst, src

[Operation] $\text{dst} \leftarrow \text{dst} \vee \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
OR1	CY, A.bit
	CY, PSW.bit
	CY, saddr.bit
	CY, sfr.bit
	CY, [HL].bit
	CY, ES:[HL].bit

[Flag]

Z	AC	CY
		x

[Description]

- The bit data of the destination operand (dst) specified by the first operand is ORed with that of the source operand (src) specified by the second operand, and the result is stored in the destination operand (dst).
- The operation result is stored in the CY flag because it is specified as dst.

[Description example]

OR1 CY, P2.5; The value of the CY flag is ORed with bit 5 of port 2, and the result is stored in the CY flag.

XOR1

Exclusive Or Single Bit
1 Bit Data Exclusive Logical Sum

[Instruction format] XOR1 dst, src

[Operation] $\text{dst} \leftarrow \text{dst} \vee \text{src}$

[Operand]

Mnemonic	Operand (dst, src)
XOR1	CY, A.bit
	CY, PSW.bit
	CY, saddr.bit
	CY, sfr.bit
	CY, [HL].bit
	CY, ES:[HL].bit

[Flag]

Z	AC	CY
		x

[Description]

- The bit data of the destination operand (dst) specified by the first operand is XORed with that of the source operand (src) specified by the second operand, and the result is stored in the destination operand (dst).
- The operation result is stored in the CY flag because it is specified as dst.

[Description example]

XOR1 CY, A.7; The value of the CY flag is XORed with bit 7 of the A register, and the result is stored in the CY flag.

SET1

Set Single Bit (Carry Flag)
1 Bit Data Set

[Instruction format] SET1 dst

[Operation] $\text{dst} \leftarrow 1$

[Operand]

Mnemonic	Operand (dst)
SET1	A.bit
	PSW.bit
	!addr16.bit
	ES:!addr16.bit
	saddr.bit
	sfr.bit
	[HL].bit
	ES:[HL].bit
	CY

[Flag]

dst = PSW.bit

Z	AC	CY
x	x	x

dst = CY

Z	AC	CY
		1

In all other cases

Z	AC	CY

[Description]

- The destination operand (dst) is set to 1.
- When CY or PSW.bit is specified as the destination operand (dst), only the corresponding flag is set to 1.
- No interrupt requests are acknowledged until the instruction following SET1 PSW.bit has been executed.

[Description example]

SET1 FFE55H.1; Bit 1 of FFE55H is set to 1.

CLR1

Clear Single Bit (Carry Flag)
1 Bit Data Clear

[Instruction format] CLR1 dst

[Operation] dst ← 0

[Operand]

Mnemonic	Operand (dst)
CLR1	A.bit
	PSW.bit
	!addr16.bit
	ES:!addr16.bit
	saddr.bit
	sfr.bit
	[HL].bit
	ES:[HL].bit
	CY

[Flag]

Z	AC	CY
x	x	x

Z	AC	CY
		0

Z	AC	CY

[Description]

- The destination operand (dst) is cleared to 0.
- When CY or PSW.bit is specified as the destination operand (dst), only the corresponding flag is cleared to 0.
- No interrupt requests are acknowledged until the instruction following CLR1 PSW.bit has been executed.

[Description example]

CLR1 P3.7; Bit 7 of port 3 is cleared to 0.

NOT1

Not Single Bit (Carry Flag)
1 Bit Data Logical Negation

[Instruction format] NOT1 dst

[Operation] $\text{dst} \leftarrow \overline{\text{dst}}$

[Operand]

Mnemonic	Operand (dst)
NOT1	CY

[Flag]

Z	AC	CY
		x

[Description]

- The CY flag is inverted.

[Description example]

NOT1 CY; The CY flag is inverted.

6.10 Call and Return Instructions

The following instructions are used to call a subroutine or interrupt servicing routine and return from it.

- CALL
- CALLT
- BRK
- RET
- RETI
- RETB

CALLCall
Subroutine Call

[Instruction format] CALL target

[Operation]

$$\begin{aligned} (SP - 2) &\leftarrow (PC + n)s, \\ (SP - 3) &\leftarrow (PC + n)H, \\ (SP - 4) &\leftarrow (PC + n)L, \\ SP &\leftarrow SP - 4, \\ PC &\leftarrow \text{target} \end{aligned}$$

Remark n is 4 when using !!addr20, 3 when using !addr16 or \$!addr20, and 2 when using AX, BC, DE, or HL.

[Operand]

Mnemonic	Operand (target)
CALL	AX
	BC
	DE
	HL
	\$!addr20
	!addr16
	!!addr20

[Flag]

Z	AC	CY

[Description]

- This instruction calls a subroutine using a 20- or 16-bit absolute address or a register indirect address.
- The start address of the next instruction ($PC + n$) is saved on the stack, and program execution branches to the address specified by the target operand (target).

[Description example]

CALL !3E000H; A subroutine call is made to address 3E000H.

CALLT

Call Table

Subroutine Call (Refer to the Call Table)

[Instruction format] CALLT [addr5]

[Operation]

$$\begin{aligned} (SP - 2) &\leftarrow (PC + 2)_S, \\ (SP - 3) &\leftarrow (PC + 2)_H, \\ (SP - 4) &\leftarrow (PC + 2)_L, \\ PCs &\leftarrow 0000, \\ PCH &\leftarrow (0000, \text{addr5} + 1), \\ PCL &\leftarrow (0000, \text{addr5}), \\ SP &\leftarrow SP - 4 \end{aligned}$$

[Operand]

Mnemonic	Operand ([addr5])
CALLT	[addr5]

[Flag]

Z	AC	CY

[Description]

- This instruction calls a subroutine by referencing the call table.
- The start address of the next instruction ($PC + 2$) is saved on the stack, and program execution branches to the address indicated by the word data in the call table. Specify an even call table address in the range from 00080H to 000BFH, with the four higher-order bits fixed to 0000B and the 16 lower-order bits indicated by addr5.

[Description example]

CALLT [80H]; A subroutine call is made to the address stored as word data at 00080H and 00081H.

[Remark]

Only specify even addresses. addr5 is immediate data or a label that specifies a 16-bit even address in the range from 0080H to 00BFH. In the address specified by addr5, bits 15 to 6 are fixed to 0000000010B, bit 0 is fixed to 0B, and bits 5 to 1 are variable.

BRK

Break
Software Vectored Interrupt

[Instruction format] BRK

[Operation]

$$\begin{aligned} (SP - 1) &\leftarrow PSW, \\ (SP - 2) &\leftarrow (PC + 2)_S, \\ (SP - 3) &\leftarrow (PC + 2)_H, \\ (SP - 4) &\leftarrow (PC + 2)_L, \\ PCs &\leftarrow 0000, \\ PCH &\leftarrow (0007FH), \\ PCL &\leftarrow (0007EH), \\ SP &\leftarrow SP - 4, \\ IE &\leftarrow 0 \end{aligned}$$

[Operand]
None

[Flag]

Z	AC	CY

[Description]

- This instruction generates a software interrupt.
- The PSW and the start address of the next instruction (PC + 2) are saved on the stack, and the IE flag is cleared to 0. Program execution branches to the address specified by the word data stored at vector addresses 0007EH and 0007FH. Subsequent maskable vectored interrupts are disabled because the IE flag is 0.
- Use the RETB instruction to return from the software vectored interrupt generated with this instruction.

RETReturn
Return from Subroutine

[Instruction format] RET

[Operation] PCL $\leftarrow$ (SP),
 PCH $\leftarrow$ (SP + 1),
 PCs $\leftarrow$ (SP + 2),
 SP $\leftarrow$ SP + 4

[Operand]
 None

[Flag]

Z	AC	CY

[Description]

- This instruction causes program execution to return from a subroutine called by a CALL or CALLT instruction.
- The word data saved on the stack are restored to the PC, and program execution returns from the subroutine.

RETI

Return from Interrupt
Return from Hardware Vectored Interrupt

[Instruction format] RETI

[Operation]

$$\begin{aligned} \text{PCL} &\leftarrow (\text{SP}), \\ \text{PCH} &\leftarrow (\text{SP} + 1), \\ \text{PCs} &\leftarrow (\text{SP} + 2), \\ \text{PSW} &\leftarrow (\text{SP} + 3), \\ \text{SP} &\leftarrow \text{SP} + 4 \end{aligned}$$

[Operand]

None

[Flag]

Z	AC	CY
R	R	R

[Description]

- This instruction causes program execution to return from a vectored interrupt.
- The data saved on the stack are restored to the PC and PSW, and program execution returns from the interrupt servicing routine.
- This instruction cannot be used to return from a software interrupt generated by the BRK instruction.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Caution]

Be sure to use the RETI instruction for restoring from the non-maskable interrupt.

RETB

Return from Break
Return from Software Vectored Interrupt

[Instruction format] RETB

[Operation]

$$\begin{aligned} \text{PCL} &\leftarrow (\text{SP}), \\ \text{PCH} &\leftarrow (\text{SP} + 1), \\ \text{PCs} &\leftarrow (\text{SP} + 2), \\ \text{PSW} &\leftarrow (\text{SP} + 3), \\ \text{SP} &\leftarrow \text{SP} + 4 \end{aligned}$$

[Operand]

None

[Flag]

Z	AC	CY
R	R	R

[Description]

- This instruction causes program execution to return a software interrupt generated by the BRK instruction.
- The data saved on the stack are restored to the PC and PSW, and program execution returns from the interrupt servicing routine.
- No interrupt requests are acknowledged until the next instruction has been executed.

6.11 Stack Manipulation Instructions

The following instructions are used to manipulate the stack, including the stack pointer (SP).

- PUSH
- POP
- MOVW SP, src MOVW rp, SP
- ADDW SP, #byte
- SUBW SP, #byte

PUSHPush
Push

[Instruction format] PUSH src

[Operation]	When src = rp	When src = PSW
	(SP - 1) ← rpH,	(SP - 1) ← PSW,
	(SP - 2) ← rpL,	(SP - 2) ← 00H,
	SP ← SP - 2	SP ← SP - 2

[Operand]

Mnemonic	Operand (src)
PUSH	PSW
	rp

[Flag]

Z	AC	CY

[Description]

- The data of the register specified by the source operand (src) are saved to the stack.

[Description example]

PUSH AX; The content of the AX register is saved on the stack.

POPPop
Pop

[Instruction format] POP dst

[Operation]

When dst = rp	When dst = PSW
rpL $\leftarrow$ (SP),	PSW $\leftarrow$ (SP + 1),
rpH $\leftarrow$ (SP + 1),	SP $\leftarrow$ SP + 2
SP $\leftarrow$ SP + 2	

[Operand]

Mnemonic	Operand (dst)
POP	PSW
	rp

[Flag]

dst = rp

Z	AC	CY

dst = PSW

Z	AC	CY
R	R	R

[Description]

- The data on the stack are restored to the register specified by the destination operand (dst).
- When PSW is specified as the operand, each flag is replaced with the data on the stack.
- No interrupt requests are acknowledged until the instruction following POP PSW has been executed.

[Description example]

POP AX; The data on the stack are restored to the AX register.

MOVW SP, src
MOVW rp, SP

Move Word
 Word Data Transfer with Stack Pointer

[Instruction format] MOVW dst, src

[Operation] dst ← src

[Operand]

Mnemonic	Operand (dst, src)
MOVW	SP, #word
	SP, AX
	AX, SP
	HL, SP
	BC, SP
	DE, SP

[Flag]

Z	AC	CY

[Description]

- This instruction manipulates the content of the stack pointer.
- The content of the source operand (src), specified by the second operand, is stored in the destination operand (dst), specified by the first operand.

[Description example]

MOVW SP, #FE20H; FE20H is stored in the stack pointer.

ADDW SP, #byte

Add stack pointer
Addition of Stack Pointer

[Instruction format] ADDW SP, src

[Operation] $SP \leftarrow SP + src$

[Operand]

Mnemonic	Operand (src)
ADDW	SP, #byte

[Flag]

Z	AC	CY

[Description]

- The stack pointer specified by the first operand and the source operand (src) specified by the second operand are added and the result is stored in the stack pointer.

[Description example]

ADDW SP, #12H; Stack pointer and 12H are added, and the result is stored in the stack pointer.

SUBW SP, #byte

Sub stack pointer
Subtraction of Stack Pointer

[Instruction format] SUBW SP, src

[Operation] $SP \leftarrow SP - src$

[Operand]

Mnemonic	Operand (src)
SUBW	SP, #byte

[Flag]

Z	AC	CY

[Description]

- Source operand (src) specified by the second operand is subtracted from the stack pointer specified by the first operand, and the result is stored in the stack pointer.

[Description example]

SUBW SP, #12H; 12H is subtracted from the stack pointer, and the result is stored in the stack pointer.

6.12 Unconditional Branch Instruction

The following instruction is used to branch unconditionally.

- BR

BRBranch
Unconditional Branch

[Instruction format] BR target

[Operation] PC ← target

[Operand]

Mnemonic	Operand (target)
BR	AX
	\$addr20
	\$!addr20
	!addr16
	!!addr20

[Flag]

Z	AC	CY

[Description]

- This instruction causes an unconditional branch.
- The word data of the target address operand (target) are transferred to the PC, and program execution branches unconditionally to the address specified by the target operand.

[Description example]

BR !!12345H; Program execution branches to address 12345H.

6.13 Conditional Branch Instructions

The following instructions are used to branch conditionally.

- BC
- BNC
- BZ
- BNZ
- BH
- BNH
- BT
- BF
- BTCLR

BC

Branch if Carry

Conditional Branch with Carry Flag (CY = 1)

[Instruction format] BC \$addr20

[Operation] $PC \leftarrow PC + 2 + \text{jdisp8}$ if CY = 1

[Operand]

Mnemonic	Operand (\$addr20)
BC	\$addr20

[Flag]

Z	AC	CY

[Description]

- If the value of the CY flag is 1, program execution branches to the address specified by the operand. Otherwise, execution proceeds to the next instruction.

[Description example]

BC \$00300H; If the value of the CY flag is 1, program execution branches to address 00300H. This instruction starts at an address in the range from 0027FH to 0037EH.

BNC

Branch if Not Carry
Conditional Branch with Carry Flag (CY = 0)

[Instruction format] BNC \$addr20

[Operation] $PC \leftarrow PC + 2 + \text{jdisp8}$ if CY = 0

[Operand]

Mnemonic	Operand (\$addr20)
BNC	\$addr20

[Flag]

Z	AC	CY

[Description]

- If the value of the CY flag is 0, program execution branches to the address specified by the operand. Otherwise, execution proceeds to the next instruction.

[Description example]

BNC \$00300H; If the value of the CY flag is 0, program execution branches to address 00300H. This instruction starts at an address in the range from 0027FH to 0037EH.

BZ

Branch if Zero

Conditional Branch with Zero Flag (Z = 1)

[Instruction format] BZ \$addr20

[Operation] $PC \leftarrow PC + 2 + \text{jdisp8}$ if Z = 1

[Operand]

Mnemonic	Operand (\$addr20)
BZ	\$addr20

[Flag]

Z	AC	CY

[Description]

- If the value of the Z flag is 1, program execution branches to the address specified by the operand. Otherwise, execution proceeds to the next instruction.

[Description example]

DEC B

BZ \$003C5H; If the value of the B register is 0, program execution branches to address 003C5H. This instruction starts at an address in the range from 00344H to 00443H.

BNZ

Branch if Not Zero
Conditional Branch with Zero Flag (Z = 0)

[Instruction format] BNZ \$addr20

[Operation] $PC \leftarrow PC + 2 + \text{jdisp8}$ if Z = 0

[Operand]

Mnemonic	Operand (\$addr20)
BNZ	\$addr20

[Flag]

Z	AC	CY

[Description]

- If the value of the Z flag is 0, program execution branches to the address specified by the operand. Otherwise, execution proceeds to the next instruction.

[Description example]

CMP A, #55H

BNZ \$00A39H; If the value of the A register is not 55H, program execution branches to address 00A39H. This instruction starts at an address in the range from 009B8H to 00AB7H.

BH

Branch if Higher than

Conditional branch by numeric value comparison ((Z ∨ CY) = 0)

[Instruction format] BH \$addr20

[Operation] $PC \leftarrow PC + 3 + \text{jdisp8}$ if $(Z \vee CY) = 0$

[Operand]

Mnemonic	Operand (\$addr20)
BH	\$addr20

[Flag]

Z	AC	CY

[Description]

- If the logical OR of the Z and CY flags is 0, program execution branches to the address specified by the operand. Otherwise, execution proceeds to the next instruction.
- This instruction is used to compare unsigned values. It checks whether the value of the first operand of the immediately preceding CMP instruction is greater than that of the second operand.

[Description example]

CMP A, C

BH \$00356H; If the content of the A register is greater than that of the C register, program execution branches to address 00356H. This instruction starts at an address in the range from 002D4H to 003D3H.

BNH

Branch if Not Higher than

Conditional branch by numeric value comparison ((Z ∨ CY) = 1)

[Instruction format] BNH \$addr20

[Operation] $PC \leftarrow PC + 3 + \text{jdisp8}$ if $(Z \vee CY) = 1$

[Operand]

Mnemonic	Operand (\$addr20)
BNH	\$addr20

[Flag]

Z	AC	CY

[Description]

- If the logical OR of the Z and CY flags is 1, program execution branches to the address specified by the operand. Otherwise, execution proceeds to the next instruction.
- This instruction is used to compare unsigned values. It checks whether the value of the first operand of the immediately preceding CMP instruction is less than or equal to that of the second operand.

[Description example]

CMP A, C

BNH \$00356H; If the content of the A register is less than or equal to that of the C register, program execution branches to address 00356H. This instruction starts at an address in the range from 002D4H to 003D3H.

BT

Branch if True

Conditional Branch by Bit Test (Byte Data Bit = 1)

[Instruction format] BT bit, \$addr20

[Operation] $PC \leftarrow PC + b + jdisp8$ if bit = 1

[Operand]

Mnemonic	Operand (bit, \$addr20)	b (Number of bytes)
BT	saddr.bit, \$addr20	4
	sfr.bit, \$addr20	4
	A.bit, \$addr20	3
	PSW.bit, \$addr20	4
	[HL].bit, \$addr20	3
	ES:[HL].bit, \$addr20	4

[Flag]

Z	AC	CY

[Description]

- If the value of the first operand (bit) is 1, program execution branches to the address specified by the second operand (\$addr20). Otherwise, execution proceeds to the next instruction.

[Description example]

BT FFE47H.3, \$0055CH; If bit 3 at address FFE47H is 1, program execution branches to address 0055CH. This instruction starts at an address in the range from 004DAH to 005D9H.

BF

Branch if False

Conditional Branch by Bit Test (Byte Data Bit = 0)

[Instruction format] BF bit, \$addr20

[Operation] $PC \leftarrow PC + b + jdisp8$ if bit = 0

[Operand]

Mnemonic	Operand (bit, \$addr20)	b (Number of bytes)
BF	saddr.bit, \$addr20	4
	sfr.bit, \$addr20	4
	A.bit, \$addr20	3
	PSW.bit, \$addr20	4
	[HL].bit, \$addr20	3
	ES:[HL].bit, \$addr20	4

[Flag]

Z	AC	CY

[Description]

- If the value of the first operand (bit) is 0, program execution branches to the address specified by the second operand (\$addr20). Otherwise, execution proceeds to the next instruction.

[Description example]

BF P2.2, \$01549H; If bit 2 of port 2 is 0, program execution branches to address 01549H. This instruction starts at an address in the range from 014C6H to 015C5H.

BTCLR

Branch if True and Clear

Conditional Branch and Clear by Bit Test (Byte Data Bit =1)

[Instruction format] BTCLR bit, \$addr20

[Operation] $PC \leftarrow PC + b + jdisp8$ if bit = 1, then bit $\leftarrow 0$

[Operand]

Mnemonic	Operand (bit, \$addr20)	b (Number of bytes)
BTCLR	saddr.bit, \$addr20	4
	sfr.bit, \$addr20	4
	A.bit, \$addr20	3
	PSW.bit, \$addr20	4
	[HL].bit, \$addr20	3
	ES:[HL].bit, \$addr20	4

[Flag]

bit = PSW.bit

Z	AC	CY
x	x	x

In all other cases

Z	AC	CY

[Description]

- If the value of the first operand (bit) is 1, it is cleared to 0 and program execution branches to the address specified by the second operand (\$addr20). Otherwise, execution proceeds to the next instruction.
- When PSW.bit is specified as the first operand (bit), the corresponding flag is cleared to 0.
- No interrupt requests are acknowledged until the instruction following BTCLR PSW.bit, \$addr20 has been executed.

[Description example]

BTCLR PSW.0, \$00356H; If bit 0 (CY flag) of PSW is 1, it is cleared to 0, and program execution branches to address 00356H. This instruction starts at an address in the range from 002D4H to 003D3H.

6.14 Conditional Skip Instructions

The following instructions are used to conditionally skip the next instruction.

- SKC
- SKNC
- SKZ
- SKNZ
- SKH
- SKNH

SKC

Skip if CY
Skip with Carry Flag (CY = 1)

[Instruction format] SKC

[Operation] Next instruction skip if CY = 1

[Operand]
None

[Flag]

Z	AC	CY

[Description]

- When the CY flag is 1, the next instruction is skipped; it is treated as a NOP and takes one clock cycle to execute. However, if the next instruction is prefixed with ES:, it takes two clock cycles to execute.
- When the CY flag is 0, the next instruction is executed.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Description example]

MOV A, #55H

SKC

ADD A, #55H; When the value of the CY flag is 1, the value of the A register is AAH; when it is 0, the value is 55H.

SKNC

Skip if not CY
Skip with Carry Flag (CY = 0)

[Instruction format] SKNC

[Operation] Next instruction skip if CY = 0

[Operand]
None

[Flag]

Z	AC	CY

[Description]

- When the CY flag is 0, the next instruction is skipped; it is treated as a NOP and takes one clock cycle to execute. However, if the next instruction is prefixed with ES:, it takes two clock cycles to execute.
- When the CY flag is 1, the next instruction is executed.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Description example]

MOV A, #55H

SKNC

ADD A, #55H; When the value of the CY flag is 1, the value of the A register is AAH; when it is 0, the value is 55H.

SKZSkip if Z
Skip with Zero Flag (Z = 1)

[Instruction format] SKZ

[Operation] Next instruction skip if Z = 1

[Operand]
None

[Flag]

Z	AC	CY

[Description]

- When the Z flag is 1, the next instruction is skipped; it is treated as a NOP and takes one clock cycle to execute. However, if the next instruction is prefixed with ES:, it takes two clock cycles to execute.
- When the Z flag is 0, the next instruction is executed.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Description example]

MOV A, #55H

SKZ

ADD A, #55H; When the value of the Z flag is 0, the value of the A register is AAH; when it is 1, the value is 55H.

SKNZ

Skip if not Z
Skip with Zero Flag (Z = 0)

[Instruction format] SKNZ

[Operation] Next instruction skip if Z = 0

[Operand]
None

[Flag]

Z	AC	CY

[Description]

- When the Z flag is 0, the next instruction is skipped; it is treated as a NOP and takes one clock cycle to execute. However, if the next instruction is prefixed with ES:, it takes two clock cycles to execute.
- When the Z flag is 1, the next instruction is executed.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Description example]

MOV A, #55H

SKNZ

ADD A, #55H; When the value of the Z flag is 1, the value of the A register is AAH; when it is 0, the value is 55H.

SKH

Skip if Higher than
Skip with numeric value comparison ((Z ∨ CY) = 0)

[Instruction format] SKH

[Operation] Next instruction skip if (Z ∨ CY) = 0

[Operand]
None

[Flag]

Z	AC	CY

[Description]

- When the logical OR of Z and CY is 0, the next instruction is skipped; it is treated as a NOP and takes one clock cycle to execute. However, if the next instruction is prefixed with ES:, it takes two clock cycles to execute.
- When the logical OR of Z and CY is 1, the next instruction is executed.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Description example]

CMP A, #80H

SKH

CALL !!TARGET; When the value of the A register is greater than 80H, the CALL instruction is skipped, and the next instruction is executed. When the value of the A register is less than or equal to 80H, the CALL instruction is executed, and program execution branches to the target address.

SKNH

Skip if Not Higher than

Skip with numeric value comparison ((Z ∨ CY) = 1)

[Instruction format] SKNH

[Operation] Next instruction skip if (Z ∨ CY) = 1

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- When the logical OR of Z and CY is 1, the next instruction is skipped; it is treated as a NOP and takes one clock cycle to execute. However, if the next instruction is prefixed with ES:, it takes two clock cycles to execute.
- When the logical OR of Z and CY is 0, the next instruction is executed.
- No interrupt requests are acknowledged until the next instruction has been executed.

[Description example]

CMP A, #80H

SKNH

CALL !!TARGET; When the value of the A register is less than or equal to 80H, the CALL instruction is skipped, and the next instruction is executed. When the value of the A register is greater than 80H, the CALL instruction is executed, and program execution branches to the target address.

6.15 CPU Control Instructions

The following instructions are used to control the operation of the CPU itself.

- SEL RBn
- NOP
- EI
- DI
- HALT
- STOP

Caution The instruction set of the RL78-S1 core does not include the following CPU control instruction.

- SEL RBn (Register Bank Selection)

SEL RBn

Select Register Bank
Register Bank Selection

[Instruction format] SEL RBn

[Operation] RBS0, RBS1 $\leftarrow$ n; (n = 0 to 3)

[Operand]

Mnemonic	Operand (RBn)
SEL	RBn

[Flag]

Z	AC	CY

[Description]

- The register bank specified by the operand (RBn) is used for the next and subsequent instructions.
- RBn ranges from RB0 to RB3.

[Description example]

SEL RB2; Register bank 2 is selected as the register bank for use by the next and subsequent instructions.

NOPNo Operation
No Operation

[Instruction format] NOP

[Operation] No Operation

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- No operation is executed by this instruction, but it still requires its specified execution time.

EI	Enable Interrupt Interrupt Enabled
-----------	---------------------------------------

[Instruction format] EI

[Operation] $IE \leftarrow 1$

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- Maskable interrupts are enabled, and the interrupt enable flag (IE) is set to 1.
- No interrupt requests are acknowledged until the next instruction has been executed.
- If this instruction is executed, vectored interrupt acknowledgment from another source can be disabled. For details, refer to the description of interrupt functions in the user's manual for each product.

DI	Disable Interrupt Interrupt Disabled
-----------	---

[Instruction format] DI

[Operation] $IE \leftarrow 0$

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- Maskable interrupt acknowledgment by vectored interrupt is disabled, and the interrupt enable flag (IE) is cleared to 0.
- No interrupt requests are acknowledged until the next instruction has been executed.
- For details of interrupt servicing, refer to the description of interrupt functions in the user's manual for each product.

HALTHalt
HALT Mode Set

[Instruction format] HALT

[Operation] Set HALT Mode

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- This instruction is used to set the HALT mode to stop the CPU operation clock. The total power consumption of the system can be decreased with intermittent operation by combining this mode with the normal operation mode.

STOPStop
Stop Mode Set

[Instruction format] STOP

[Operation] Set STOP Mode

[Operand]

None

[Flag]

Z	AC	CY

[Description]

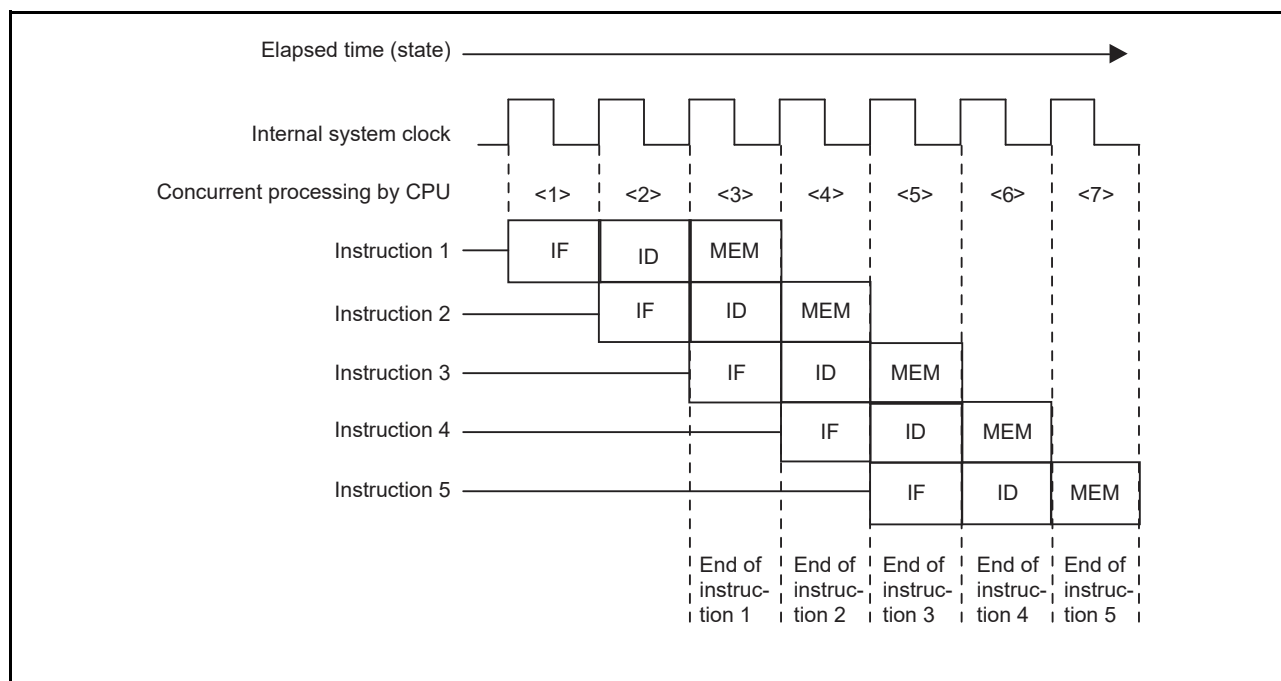
- This instruction is used to set the STOP mode to stop the main system clock oscillator and to stop the whole system. Power consumption can be minimized to only leakage current.

Section 7 Pipeline

7.1 Features

The RL78 microcontroller uses three-stage pipeline control to enable single-cycle execution of almost all instructions. Instructions are executed in three stages: instruction fetching (IF), instruction decoding (ID), and memory access (MEM).

Figure 7 - 1 Example of Pipeline Execution of Five Typical Instructions



- IF (instruction fetching): The instruction is fetched and the fetch pointer is incremented.
- ID (instruction decoding): The instruction is decoded and the address is calculated.
- MEM (memory access): The decoded instruction is executed and access to memory at the target address proceeds.

7.2 Numbers of Operating Clock Cycles

The RL78 microcontroller operates with a fixed number of clock cycles for each instruction, allowing the estimation of program execution times. The numbers of operating clock cycles are listed in **5.5 List of Operations**, except for the cases described in 7.2.1, 7.2.2, and 7.2.3.

7.2.1 Access to flash memory contents as data

When the contents of the flash memory are accessed as data, the pipeline operation is stopped at the MEM stage. Therefore, the numbers of clock cycles for operations are increased from the listed numbers of clock cycles. For details, see **5.5 List of Operations**.

7.2.2 Fetching instructions from RAM

Fetching an instruction from RAM takes more clock cycles than it takes to execute the instruction, so the CPU is not capable of fast enough access to the instructions. The CPU therefore waits until the instruction queue is filled. If a fetched instruction involves data access to RAM, the CPU also has to wait. For the RL78-S2 and RL78-S3 cores, fetching an instruction from the internal RAM area takes up to $(2 \times N + 3)$ clock cycles, where N is the number of clock cycles required when fetching an instruction from the internal ROM (flash memory) area. For the RL78-S1 core, fetching an instruction from the internal RAM area takes up to $(4 \times N + 6)$ clock cycles.

7.2.3 Hazards related to combinations of instructions

In the case of indirect access to the data contained by a register immediately after the writing to the register that is to be used for the indirect access, a one- to two-clock wait is inserted for the RL78-S1 core, and a one-clock wait is inserted for the RL78-S2 core and the RL78-S3 core.

Register Name	Previous Instruction	Next Instruction Operand (or Instruction)
DE	Write instruction to D register ^{Note 1} Write instruction to E register ^{Note 1} Write instruction to DE register ^{Note 1} SEL RB _n ^{Note 2}	[DE], [DE+byte]
HL	Write instruction to H register ^{Note 1} Write instruction to L register ^{Note 1} Write instruction to HL register ^{Note 1} SEL RB _n ^{Note 2}	[HL], [HL+byte], [HL+B], [HL+C], [HL].bit
B	Write instruction to B register ^{Note 1} SEL RB _n ^{Note 2}	word[B], [HL+B]
C	Write instruction to C register ^{Note 1} SEL RB _n ^{Note 2}	word[C], [HL+C]
BC	Write instruction to B register ^{Note 1} Write instruction to C register ^{Note 1} Write instruction to BC register ^{Note 1} SEL RB _n ^{Note 2}	word[BC], [HL+B], [HL+C]
SP	MOVW SP, #word MOVW SP, AX ADDW SP, #byte SUBW SP, #byte	[SP+byte] CALL instruction, CALLT instruction, BRK instruction, SOFT instruction, RET instruction, RETI instruction, RETB instruction, interrupt, PUSH instruction, POP instruction
CS	MOV CS, #byte MOV CS, A	CALL rp BR AX
AX	Write instruction to A register ^{Note 1} Write instruction to X register ^{Note 1} Write instruction to AX register ^{Note 1} SEL RB _n ^{Note 2}	BR AX
AX BC DE HL	Write instruction to A register ^{Note 1} Write instruction to X register ^{Note 1} Write instruction to B register ^{Note 1} Write instruction to C register ^{Note 1} Write instruction to D register ^{Note 1} Write instruction to E register ^{Note 1} Write instruction to H register ^{Note 1} Write instruction to L register ^{Note 1} Write instruction to AX register ^{Note 1} Write instruction to BC register ^{Note 1} Write instruction to DE register ^{Note 1} Write instruction to HL register ^{Note 1} SEL RB _n ^{Note 2}	CALL rp

Note 1. Register write instructions also require wait insertion when overwriting the target register values with the use of direct addressing, short direct addressing, register indirect addressing, based addressing, or based indexed addressing.

Note 2. The instruction set of the RL78-S1 core does not include this instruction.

Appendix A Instruction Index (Mnemonic by Function)

Table A - 1 List of Instructions Classified by Function (1/3)

Function	Mnemonic	Description	Function in Detail (on Page)	Clock Cycle in Detail (on Page)		
				RL78-S1	RL78-S2	RL78-S3
8-bit data transfer instructions	MOV	Byte data transfer	130	39	56	73
	XCH	Byte data transfer	132	41	58	75
	ONEB	Byte data 01H set	133	42	59	76
	CLRB	Byte data clear	134	42	59	76
	MOVS	Byte data transfer and PSW change	135	42	59	76
16-bit data transfer instructions	MOVW	Word data transfer	137	42	59	76
	XCHW	Word data exchange	139	44	61	78
	ONEW	Word data 0001H set	140	44	61	78
	CLRW	Word data clear	141	44	61	78
8-bit operation instructions	ADD	Byte data addition	143	44	61	78
	ADDC	Addition of byte data with carry	144	45	62	79
	SUB	Byte data subtraction	145	45	62	79
	SUBC	Subtraction of byte data with carry	146	46	63	80
	AND	Logical product of byte data	147	46	63	80
	OR	Logical sum of byte data	148	47	64	81
	XOR	Exclusive logical sum of byte data	149	47	64	81
	CMP	Byte data comparison	150	48	65	82
	CMP0	Byte data zero comparison	151	48	65	82
	CMPS	Byte data comparison	152	48	65	82
16-bit operation instructions	ADDW	Word data addition	154	49	66	83
	SUBW	Word data subtraction	155	49	66	83
	CMPW	Word data comparison	156	49	66	83
Multiply, divide, and multiply-and-accumulate instructions	MULU	Unsigned multiplication of data	158	49	66	84
	MULHU	Unsigned multiplication of data	159	—	—	84
	MULH	Signed multiplication of data	160	—	—	84
	DIVHU	Unsigned division of data	161	—	—	84
	DIVWU	Unsigned division of data	162	—	—	84
	MACHU	Unsigned multiplication and accumulation of data	163	—	—	84
	MACH	Signed multiplication and accumulation of data	164	—	—	84
Increment and decrement instructions	INC	Byte data increment	166	50	67	85
	DEC	Byte data decrement	167	50	67	85
	INCW	Word data increment	168	50	67	85
	DECW	Word data decrement	169	50	67	85

Table A - 1 List of Instructions Classified by Function (2/3)

Function	Mnemonic	Description	Function in Detail (on Page)	Clock Cycle in Detail (on Page)		
				RL78-S1	RL78-S2	RL78-S3
Shift instructions	SHR	Logical shift to the right	171	50	67	85
	SHRW	Logical shift to the right	172	50	67	85
	SHL	Logical shift to the left	173	50	67	85
	SHLW	Logical shift to the left	174	50	67	85
	SAR	Arithmetic shift to the right	175	50	67	85
	SARW	Arithmetic shift to the right	176	50	67	85
Rotate instructions	ROR	Byte data rotation to the right	178	51	68	86
	ROL	Byte data rotation to the left	179	51	68	86
	RORC	Byte data rotation to the right with carry	180	51	68	86
	ROLC	Byte data rotation to the left with carry	181	51	68	86
	ROLWC	Word data rotation to the left with carry	182	51	68	86
Bit manipulation instructions	MOV1	1 bit data transfer	184	51	68	86
	AND1	1 bit data logical product	185	51	68	86
	OR1	1 bit data logical sum	186	51	68	86
	XOR1	1 bit data exclusive logical sum	187	52	69	87
	SET1	1 bit data set	188	52	69	87
	CLR1	1 bit data clear	189	52	69	87
	NOT1	1 bit data logical negation	190	52	69	87
Call and return instructions	CALL	Subroutine call	192	53	70	88
	CALLT	Subroutine call (refer to the call table)	193	53	70	88
	BRK	Software vectored interrupt	194	53	70	88
	RET	Return from subroutine	195	53	70	88
	RETI	Return from hardware vectored interrupt	196	53	70	88
	RETB	Return from software vectored interrupt	197	53	70	88
Stack manipulation instructions	PUSH	Push	199	54	71	89
	POP	Pop	200	54	71	89
	MOVW SP, src	Word data transfer with stack pointer	201	54	71	89
	MOVW rp, SP	Word data transfer with stack pointer	201	54	71	89
	ADDW SP, #byte	Addition of stack pointer	202	54	71	89
	SUBW SP, #byte	Subtraction of stack pointer	203	54	71	89
Unconditional branch instruction	BR	Unconditional branch	205	54	71	89
Conditional branch instructions	BC	Conditional branch with carry flag (CY = 1)	207	54	71	89
	BNC	Conditional branch with carry flag (CY = 0)	208	54	71	89
	BZ	Conditional branch with zero flag (Z = 1)	209	54	71	89
	BNZ	Conditional branch with zero flag (Z = 0)	210	54	71	89
	BH	Conditional branch by numeric value comparison ((Z ∨ CY) = 0)	211	54	71	89

Table A - 1 List of Instructions Classified by Function (3/3)

Function	Mnemonic	Description	Function in Detail (on Page)	Clock Cycle in Detail (on Page)		
				RL78-S1	RL78-S2	RL78-S3
Conditional branch instructions	BNH	Conditional branch by numeric value comparison ($(Z \vee CY) = 1$)	212	54	71	89
	BT	Conditional branch by bit test (byte data bit = 1)	213	54	71	89
	BF	Conditional branch by bit test (byte data bit = 0)	214	55	72	90
	BTCLR	Conditional branch and clear by bit test (byte data bit = 1)	215	55	72	90
Conditional skip instructions	SKC	Skip with carry flag ($CY = 1$)	217	55	72	90
	SKNC	Skip with carry flag ($CY = 0$)	218	55	72	90
	SKZ	Skip with zero flag ($Z = 1$)	219	55	72	90
	SKNZ	Skip with zero flag ($Z = 0$)	220	55	72	90
	SKH	Skip with numeric value comparison ($((Z \vee CY) = 0)$)	221	55	72	90
	SKNH	Skip with numeric value comparison ($((Z \vee CY) = 1)$)	222	55	72	90
CPU control instructions	SEL RBn	Register bank selection	224	—	72	90
	NOP	No operation	225	55	72	90
	EI	Interrupt enabled	226	55	72	90
	DI	Interrupt disabled	227	55	72	90
	HALT	HALT mode set	228	55	72	90
	STOP	Stop mode set	229	55	72	90

Appendix B Instruction Index (Mnemonic in Alphabetical Order)

Table B - 1 List of Instructions Classified in Alphabetical Order (1/3)

Mnemonic	Description	Function in Detail (on Page)	Clock Cycle in Detail (on Page)		
			RL78-S1	RL78-S2	RL78-S3
ADD	Byte data addition	143	44	61	78
ADDC	Addition of byte data with carry	144	45	62	79
ADDW	Word data addition	154	49	66	83
ADDW SP, #byte	Addition of stack pointer	202	54	71	89
AND	Logical product of byte data	147	46	63	80
AND1	1 bit data logical product	185	51	68	86
BC	Conditional branch with carry flag (CY = 1)	207	54	71	89
BF	Conditional branch by bit test (byte data bit = 0)	214	55	72	90
BH	Conditional branch by numeric value comparison ((Z ∨ CY) = 0)	211	54	71	89
BNC	Conditional branch with carry flag (CY = 0)	208	54	71	89
BNH	Conditional branch by numeric value comparison ((Z ∨ CY) = 1)	212	54	71	89
BNZ	Conditional branch with zero flag (Z = 0)	210	54	71	89
BR	Unconditional branch	205	54	71	89
BRK	Software vectored interrupt	194	53	70	88
BT	Conditional branch by bit test (byte data bit = 1)	213	54	71	89
BTCLR	Conditional branch and clear by bit test (byte data bit = 1)	215	55	72	90
BZ	Conditional branch with zero flag (Z = 1)	209	54	71	89
CALL	Subroutine call	192	53	70	88
CALLT	Subroutine call (refer to the call table)	193	53	70	88
CLRB	Byte data clear	134	42	59	76
CLRW	Word data clear	141	44	61	78
CLR1	1 bit data clear	189	52	69	87
CMP	Byte data comparison	150	48	65	82
CMPS	Byte data comparison	152	48	65	82
CMPW	Word data comparison	156	49	66	83
CMP0	Byte data zero comparison	151	48	65	82
DEC	Byte data decrement	167	50	67	85
DECW	Word data decrement	169	50	67	85
DI	Interrupt disabled	227	55	72	90
DIVHU	Unsigned division of data	161	—	—	84
DIVWU	Unsigned division of data	162	—	—	84
EI	Interrupt enabled	226	55	72	90
HALT	HALT mode set	228	55	72	90
INC	Byte data increment	166	50	67	85

Table B - 1 List of Instructions Classified in Alphabetical Order (2/3)

Mnemonic	Description	Function in Detail (on Page)	Clock Cycle in Detail (on Page)		
			RL78-S1	RL78-S2	RL78-S3
INCW	Word data increment	168	50	67	85
MACH	Signed multiplication and accumulation of data	164	—	—	84
MACHU	Unsigned multiplication and accumulation of data	163	—	—	84
MOV	Byte data transfer	130	39	56	73
MOVS	Byte data transfer and PSW change	135	42	59	76
MOVW	Word data transfer	137	42	59	76
MOVW rp, SP	Word data transfer with stack pointer	201	54	71	89
MOVW SP, src	Word data transfer with stack pointer	201	54	71	89
MOV1	1 bit data transfer	184	51	68	86
MULH	Signed multiplication of data	160	—	—	84
MULHU	Unsigned multiplication of data	159	—	—	84
MULU	Unsigned multiplication of data	158	49	66	84
NOP	No operation	225	55	72	90
NOT1	1 bit data logical negation	190	52	69	87
ONEB	Byte data 01H set	133	42	59	76
ONEW	Word data 0001H set	140	44	61	78
OR	Logical sum of byte data	148	47	64	81
OR1	1 bit data logical sum	186	51	68	86
POP	Pop	200	54	71	89
PUSH	Push	199	54	71	89
RET	Return from subroutine	195	53	70	88
RETB	Return from software vectored interrupt	197	53	70	88
RETI	Return from hardware vectored interrupt	196	53	70	88
ROL	Byte data rotation to the left	179	51	68	86
ROLC	Byte data rotation to the left with carry	181	51	68	86
ROLWC	Word data rotation to the left with carry	182	51	68	86
ROR	Byte data rotation to the right	178	51	68	86
RORC	Byte data rotation to the right with carry	180	51	68	86
SAR	Arithmetic shift to the right	175	50	67	85
SARW	Arithmetic shift to the right	176	50	67	85
SEL RBn	Register bank selection	224	—	72	90
SET1	1 bit data set	188	52	69	87
SHL	Logical shift to the left	173	50	67	85
SHLW	Logical shift to the left	174	50	67	85
SHR	Logical shift to the right	171	50	67	85
SHRW	Logical shift to the right	172	50	67	85
SKC	Skip with carry flag (CY = 1)	217	55	72	90
SKH	Skip with numeric value comparison ((Z ∨ CY) = 0)	221	55	72	90
SKNC	Skip with carry flag (CY = 0)	218	55	72	90

Table B - 1 List of Instructions Classified in Alphabetical Order (3/3)

Mnemonic	Description	Function in Detail (on Page)	Clock Cycle in Detail (on Page)		
			RL78-S1	RL78-S2	RL78-S3
SKNH	Skip with numeric value comparison ($(Z \vee CY) = 1$)	222	55	72	90
SKNZ	Skip with zero flag ($Z = 0$)	220	55	72	90
SKZ	Skip with zero flag ($Z = 1$)	219	55	72	90
STOP	Stop mode set	229	55	72	90
SUB	Byte data subtraction	145	45	62	79
SUBC	Subtraction of byte data with carry	146	46	63	80
SUBW	Word data subtraction	155	49	66	83
SUBW SP, #byte	Subtraction of stack pointer	203	54	71	89
XCH	Byte data transfer	132	41	58	75
XCHW	Word data exchange	139	44	61	78
XOR	Exclusive logical sum of byte data	149	47	64	81
XOR1	1 bit data exclusive logical sum	187	52	69	87

Appendix C Revision History

C.1 Major Revisions in This Edition

Page	Description	Classification
Section 1 Overview		
p.1	Table 1 - 1 Differences in Functions of the Three CPU Cores: Remark was modified.	(c)
Section 2 Memory Space		
p.2	Figure 2 - 1 Memory Map of the RL78 Microcontroller was modified.	(a)
p.3	2.2.1 Mirror area: The description was modified.	(a)
p.4	2.2.2 Vector table area: The description was modified.	(c)
p.4	2.2.3 CALLT instruction table area: The description was modified.	(c)
p.5	2.2.4 Option bytes area: The description was modified.	(c)
p.5	2.2.5 Security ID setting area: The title and description were modified.	(c)
p.6	2.5 Extended Special Function Register (2nd SFR) Area: The description was modified.	(a)
Section 3 Registers		
p.16	Figure 3 - 11 Configuration of Processor Mode Control Register (PMC) of RL78-S2 Core and RL78-S3 Core was modified.	(a)
Section 4 Addressing		
p.19	4.1.4 Register indirect addressing: The title and description were modified.	(a)
p.19	Figure 4 - 5 Outline of Register Indirect Addressing: The title was modified.	(a)
p.32	Figure 4 - 25 Example of CALL and CALLT was modified.	(a)
Section 5 Instruction Set		
p.42, p.50	Table 5 - 5 Operation List of RL78-S1 Core was modified.	(a)
p.84	Table 5 - 7 Operation List of RL78-S3 Core: Caution was modified.	(c)
Section 6 Explanation of Instructions		
p.161	DIVHU: Caution was modified.	(c)
p.162	DIVWU: Caution was modified.	(c)
p.163	MACHU: [Description] was modified.	(a)
p.164	MACH: [Description] was modified.	(a)
p.175	SAR: The description was modified.	(a)

Remark "Classification" in the above table classifies revisions as follows.

- (a): Error correction, (b): Addition/change of specifications, (c): Addition/change of description or note,
 (d): Addition/change of package, part number, or management division, (e): Addition/change of related documents

C.2 Major Revisions in This Edition

Here is the revision history of the preceding editions. Section indicates the section of each edition.

(1/2)

Edition	Description	Section
Rev.0.50	First Edition issued.	Throughout
Rev.2.00	Description changed.	Section 1 OVERVIEW
	Description changed.	Section 2 Memory Space
	3.1.1 Program counter (PC): Description changed.	Section 3 Registers
	3.1.2 Program status word (PSW): Description changed.	
	3.1.2 (3) Register bank select flags (RBS0 and RBS1): Caution added.	
	3.1.3 Stack pointer (SP): Description changed.	
	3.2 General-Purpose Registers: Description changed.	
	Table 3-2. List of General-Purpose Registers: Note added.	
	3.3 ES and CS Registers: Description changed.	
	3.4.1 Processor mode control register (PMC): Description changed.	
	4.1.1 Relative addressing: Caution deleted.	Section 4 Addressing
	4.2.3 Direct addressing: Error corrected.	
	4.2.9 Stack addressing: Description changed.	
	Description of instruction set changed.	Section 5 Instruction Set
	Table 5-1. Operand Identifiers and Description Methods: Note added.	
	Table 5-2. Symbols in "Operation" Column: Note added.	
	5.5.2 Operation List of RL78-S2 Core: Description added.	
	5.5.3 Operation List of RL78-S3 Core: Description added.	
	Table 5-8. List of Instruction Formats (12/30): Note added.	
	Table 5-8. List of Instruction Formats (30/30): Note added.	
	Table 5-10. Instruction Map (2nd MAP): Note added.	
	CMP0 (Byte Data Zero Comparison): Error in flag corrected.	Section 6 Explanation of Instructions
	6.5 Multiply/Divide/Multiply & Accumulate Instructions: Caution added.	
	6.15 CPU Control Instructions: Caution added.	
	7.2 Number of Operation Clocks: Description changed.	Section 7 Pipeline
	7.2.3 Hazards related to combined instructions: Description changed and Note added.	
	INSTRUCTION INDEX (MNEMONIC: BY FUNCTION): Format changed to a table.	Appendix A Instruction Index (Mnemonic: By Function)
	INSTRUCTION INDEX (MNEMONIC: IN ALPHABETICAL ORDER): Format changed to a table.	Appendix B Instruction Index (Mnemonic: in Alphabetical Order)
Rev.2.10	3.1.2 (3) Register bank select flags (RBS0 and RBS1): Error corrected.	Section 3 Registers
	Table 5-8. List of Instruction Formats: Error corrected.	Section 5 Instruction Set
Rev.2.20	Table 5-7. Operation List of RL78-S3 Core (12/18): Caution added.	Section 5 Instruction Set
	6.5 Multiply/Divide/Multiply & Accumulate Instructions: Caution added.	Section 6 Explanation of Instructions

(2/2)

Edition	Description	Section
Rev.2.30	Table 1-1. Functional Differences between Three CPU Cores: Modified Remark.	Section 1 Overview
	Table 5-8. List of Instruction Formats (12/30): Error corrected.	Section 5 Instruction Set
	Table 5-8. List of Instruction Formats (17/30): Error corrected.	
	6.1 8-bit Data Transfer Instructions: Added [Operand] of MOV instruction.	Section 6 Explanation of Instructions

RL78 Family User's Manual: Software

Publication Date: Rev.1.00 Jan 31, 2011
 Rev.2.40 Oct 30, 2026

Published by: Renesas Electronics Corporation

RL78 Family