To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

用户手册

78K0R 微控制器

16 位单片微控制器

指令

适用于 78K0R 微控制器

文档编号：U17792CA4V0UM00（第四版）
发行日期：2007 年 8 月 NS

[备忘录]

CMOS设备的注释

① 输入引脚处的电压波形

输入噪声或一个反射波引起的波形失真可能导致错误发生。如果由于噪声等的影响使CMOS设备的输入电压范围保持在V_{IL}（MAX）和V_{IH}（MIN）之间，设备可能出错。在输入电平固定时以及输入电平从V_{IL}（MAX）过渡到V_{IH}（MIN）时的传输期间，要防止散射噪声影响设备。

② 未使用的输入引脚的处理

CMOS设备的输入端保持开路可能导致误操作。如果一个输入引脚未被连接，则由于噪音等原因可能会产生内部输入电平，从而导致误操作。CMOS设备的操作特性与Bipolar或NMOS设备不同。CMOS设备的输入电平必须借助上拉或下拉电路固定为高电平或低电平。每一个未使用引脚都应该通过上拉/下拉电阻连接到V_{DD}或GND。如果有可能尽量定义为输出引脚。对未使用引脚的处理因设备而异，必须遵循与设备相关的规定和说明。

③ ESD防护措施

如果MOS设备周围有强电场，将会击穿氧化栅极，从而影响设备的运行。因此必须采取措施，尽可能防止静电产生。一旦有静电，必须立即释放。对环境必须有适当的控制。如果空气干燥，应当使用增湿器。建议避免使用容易产生静电的绝缘体。半导体设备的存放和运输必须使用抗静电容器、抗静电屏蔽袋或导电材料容器。所有的测试和测量工具包括工作台和工作面必须良好接地。操作员应当佩戴静电消除手带以保证良好接地。不能用手直接接触半导体设备。对于装配有半导体设备的PW板也应采取类似的静电防范措施。

④ 初始化之前的状态

在上电时MOS设备的初始状态是不确定的。在刚刚上电之后，具有复位功能的MOS设备还没有完成初始化。因此上电不能保证输出引脚的电平，I/O设置和寄存器的内容。设备在收到复位信号后才进行初始化。具有复位功能的设备在上电后必须立即进行复位操作。

⑤ 电源开关顺序

在一个设备的内部操作和外部接口使用不同的电源的情况下，按照规定，应先在接通内部电源之后再接通外部电源。当关闭电源时，按照规定，先关闭外部电源再关闭内部电源。如果电源开关顺序颠倒，可能会导致设备的内部组件过电压，产生异常电流，从而引起内部组件的误操作和性能的退化。

对于每个设备电源的正确开关顺序必须依据设备的规范说明分别进行判断。

⑥ 电源关闭状态下的输入信号

不要向没有加电的设备输入信号或提供I/O上拉电源。因为输入信号或提供I/O上拉电源将引起电流注入，从而引起设备的误操作，并产生异常电流，从而使内部组件退化。

每个设备电源关闭时的信号输入必须依据设备的规范说明分别进行判断。

这些产品、技术或软件的出口必须符合出口国的出口管理法规。禁止违反出口国的相关法律

- **本文档所登载的内容有效期截止至 2008 年 08 月，信息先于产品的生产周期发布。将来可能未经预先通知而更改。在实际进行生产设计时，请参阅各产品最新的数据表或数据手册等相关资料以获取本公司产品的最新规格。**
- 并非所有的产品和/或型号都向每个国家供应。请向本公司销售代表查询产品供应及其他信息。
- 未经本公司事先书面许可，禁止复制或转载本文件中的内容。否则因本文档所登载内容引发的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权作出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用实例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，本公司概不负责。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应同意并知晓，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。
- 本公司产品质量分为：

  “标准等级”、“专业等级”以及“特殊等级”三种质量等级。

  “特殊等级”仅适用于为特定用途而根据用户指定的质量保证程序所开发的日电电子产品。另外，各种日电电子产品的推荐用途取决于其质量等级，详见如下。用户在选用本公司的产品时，请事先确认产品的质量等级。

  “标准等级”： 计算机，办公自动化设备，通信设备，测试和测量设备，音频·视频设备，家电，加工机械以及产业用机器人。

  “专业等级”： 运输设备（汽车、火车、船舶等），交通用信号控制设备，防灾装置，防止犯罪装置，各种安全装置以及医疗设备（不包括专门为维持生命而设计的设备）。

  “特殊等级： 航空器械，宇航设备，海底中继设备，原子能控制系统，为了维持生命的医疗设备、用于维持生命的装置或系统等。

  除在本公司半导体产品的数据表或数据手册等资料中另有特别规定以外，本公司半导体产品的质量等级均为“标准等级”。如果用户希望在本公司设计意图以外使用本公司半导体产品，务必事先与本公司销售代表联系以确认本公司是否同意为该项应用提供支持。

（注）

（1） 本声明中的“本公司”是指日本电气电子株式会社（NEC Electronics Corporation）及其控股公司。

（2） 本声明中的“本公司产品”是指所有由日本电气电子株式会社开发或制造的产品或为日本电气电子株式会社（定义如上）开发或制造的产品。

M5 02.11-1

前言

读者对象

本手册适用于那些希望了解 78K0R 产品功能，并设计开发相关应用系统和程序的用户。

目的

本手册用于帮助用户了解 78K0R 系列产品各种指令的功能。

组织

本手册主要分为以下部分：

- CPU 功能
- 指令集
- 指令描述

如何阅读本手册

在阅读本手册前，读者应掌握电子工程、逻辑电路和微控制器等方面的一般知识。

- 如何得到已知助记符指令功能的详细信息：
  →请参见 **附录 A 和 B。**

- 如何得到未知助记符但是了解其整体功能指令功能的详细信息：
  →参见**第五章 指令集**中的助记符，然后是**第六章 指令描述**中的功能。

- 如何从整体上理解78K0R系列产品指令的所有功能：
  →根据**目录**顺序阅读手册。标记“<R>”的地方表示为主要修改点。通过在PDF文件中的“查找框”中拷贝“<R>”可以容易的找到修改点。

- 如何获悉 78K0R 系列产品的硬件功能：
  →参考每个产品的用户手册。

约定

数据重要性：	数据的高位部分在左边，低位部分在右边
注：	文中用**注**标注的相关术语的脚注
注意事项：	需要特别关注的信息
备注：	补充信息
数的表示法：	二进制……XXXX 或 XXXXB
	十进制……..XXXX
	十六进制…XXXXH

目录

第一章 概述

1.1 与 78K0 微控制器的差异 （对于汇编用户）

（1）使用流水线减少所有指令处理的时钟周期数。现有的程序必须重新评价。

（2）所有指令码已经修改。使用汇编器重新汇编指令。当重新汇编时，由于增加了新的指令，代码大小可能会增加，但是在某些情况下如果新指令替换了新旧指令，则总体的代码大小可能会减小。

（3）存储器空间从 64KB 变为 1 MB, 堆栈区域也增加了。在汇编程序中，每当操作堆栈指针中的 RAM 内容时地址一定会改变。子程序调用嵌套和中断嵌套的层数将会增加堆栈大小。

（4）CALLT 表地址范围从“0040H 到 007FH”变为“0080H 到 00BFH”。因此，CALLT 表地址应改变。

（5）必须重写程序中用于 78K0 微控制器 bank 切换部分的汇编程序。

（6）当使用扩展 RAM 时地址发生变化。一定要改变这些地址。

（7）如果从扩展 RAM 执行指令，由于存储器空间地址已经改变，所以 BR !addr16 变为 BR !!addr20，CALL !addr16 变为 CALL !!addr20。

（8）没有 IMS 或 IXS 寄存器（用于设置存储器空间的寄存器）。如果没有使用外部存储器应删除使用这些寄存器的程序语句。如果使用外部存储器，MM/MEM 寄存器（外部存储器设置寄存器）的规范已经改变，因此请查看各产品的用户手册并改变相应的设置。

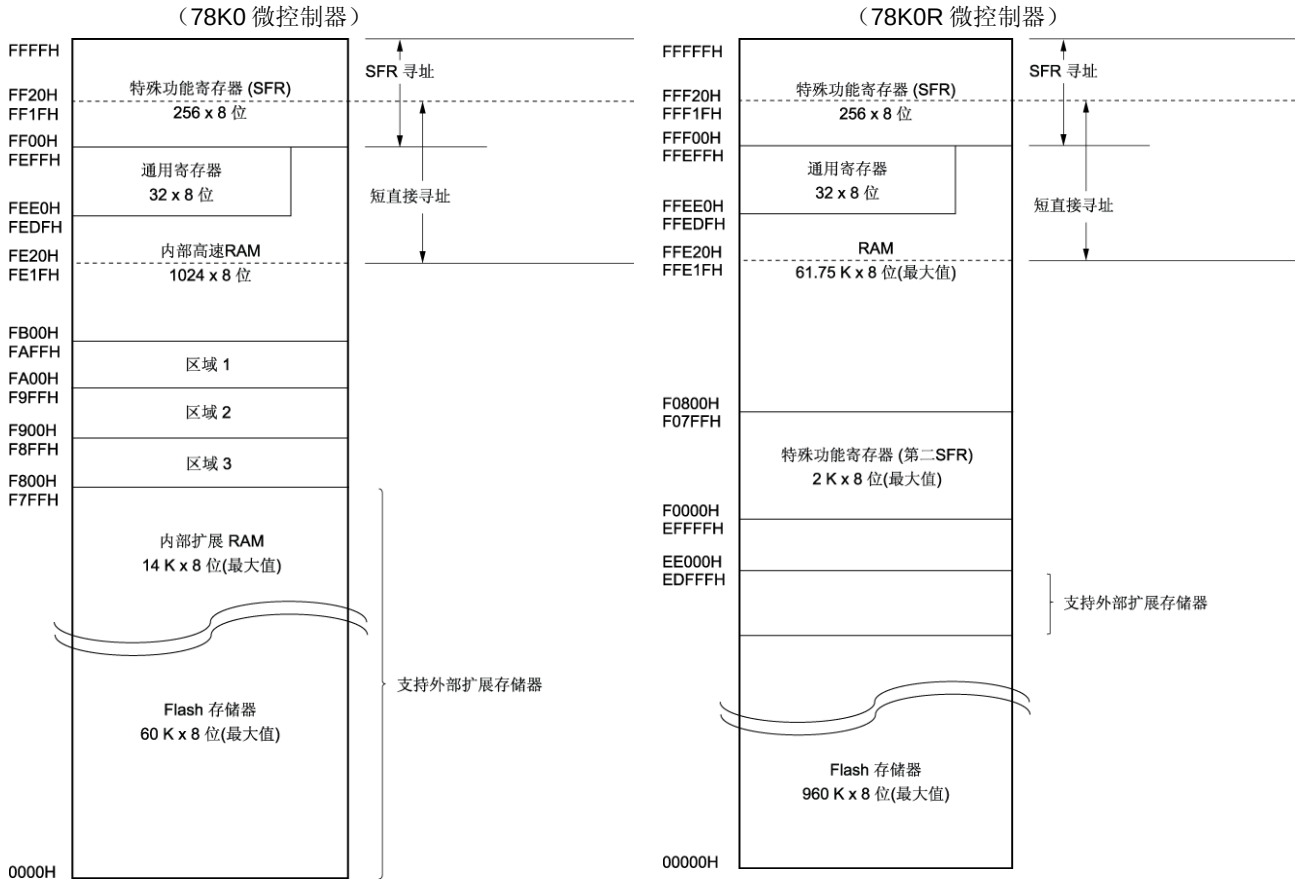
（9）如下指令已删除并有替代这些指令的代码，这样会导致代码量变大。即使使用这些指令，也会在汇编期间自动替换。

指令	操作数	备注
DIVUW	C	替代指令以移位执行除法，因此执行时间比 DIVUW 长。推荐改变此指令为增加的移位指令。
ROR4	[HL]	替代指令的执行时间比 ROR4 长。推荐改变此指令为增加的移位指令。
ROL4	[HL]	替代指令的执行时间比 ROL4 长。推荐改变此指令为增加的移位指令。
ADJBA	无	替代指令的执行时间比 ADJBA 长。无增加指令。
ADJBS	无	替代指令的执行时间比 ADJBS 长。无增加指令。
CALLF	!addr11	CALLF 自动变为 3 字节指令 CALL !addr16。 可以使用，不用修改。
DBNZ	B, $addr16 C, $addr16 saddr, $addr16	此指令分成两个：DEC B/DEC C/DEC saddr 和 BNZ $addr20。可以使用，不用修改。

2.1　存储器空间

78K0 微控制器的存储器空间仅为 64 KB，在 78K0R 微控制器中已经扩展到 1 MB。

图 2-1. 78K0 微控制器和 78K0R 微控制器的存储器映射图



- 程序存储器空间为 60 KB （最大值）。
- 内部高速 RAM 区域为 1 KB （最大值）（允许堆栈）。
  内部扩展 RAM 区域为 14 KB （最大值）（允许取值）。
- 区域 1，区域 2，和区域 3 从 F800H 到 FAFFH （固定）。
- 支持外部扩展存储器。

- 程序存储器空间为 960 KB （最大值）。
- RAM 空间为 61.75 KB （最大值）（允许堆栈，允许取值）。
- 第二 SFR 区域（名称改变）为 2 KB （最大值），从 F0000H 到 F07FFH。
- 支持外部扩展存储器。
  外部扩展存储器空间从片内 flash 存储器区域到 EDFFFH。

2.2 内部程序存储器空间

在 78K0R 微控制器中，程序存储器空间的地址范围从 00000H 到 EFFFFH。

关于内部 ROM （flash 存储器）最大容量的说明，参见各产品的用户手册。

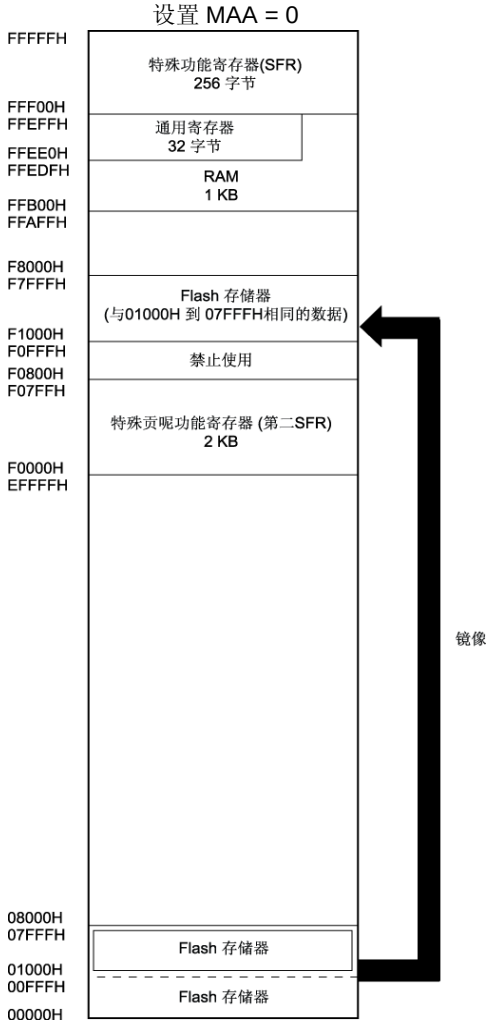
2.2.1 镜像区域

在 78K0R 微控制器中，从 00000H 到 0FFFFH （当 MAA = 0 时）和从 10000H 到 1FFFFH（当 MAA = 1 时）的数据 flash 区域镜像到从 F0000H 到 FFFFFH 的地址。通过读取从 F0000H 到 FFFFFH 中的数据可以用更少的代码实现读取数据 flash 的内容。但是在这种情况数据 flash 区域并不镜像到 SFR，扩展 SFR（第二 SFR），RAM 和禁止使用区域。

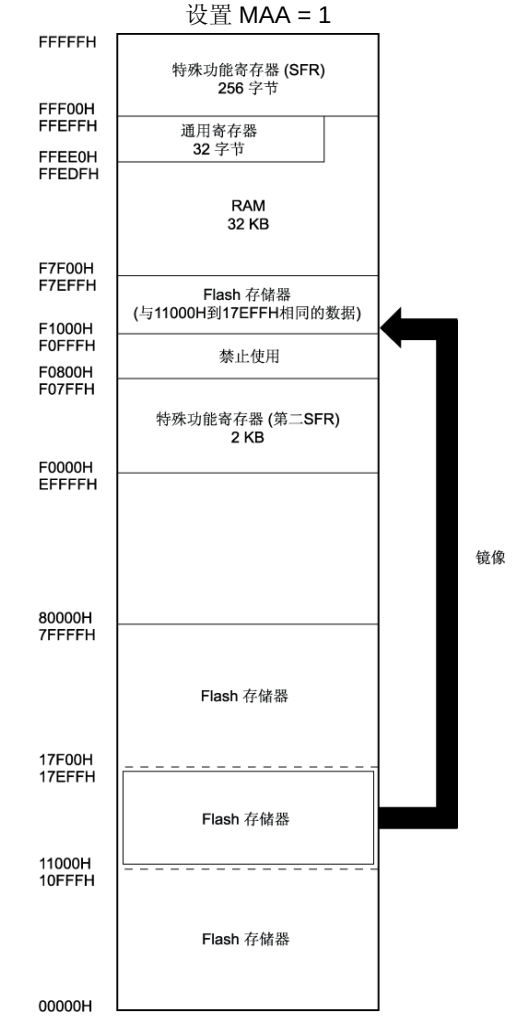
镜像区域仅能读取，禁止取指令。

示例如下。各产品规范不同，因此请参考各产品的用户手册。

例 1 （Flash 存储器：32 KB, RAM：1 KB）



例 2 （Flash 存储器：512 KB, RAM：32 KB）



备注 MAA：处理器模式控制寄存器（PMC）的第 0 位（详细情况请参考 **3.4.1 处理器模式控制寄存器（PMC）**）。

2.2.2 向量表区域

在 78K0R 微控制器中，从 0000H 到 007FH 的 128 字节区域保留为向量表区域。中断数量为 61（最大值）+RESET 向量+片上调试向量+软件中断向量。由于向量代码仅有 2 字节，中断转移目的起始地址为 64 KB，从 00000H 到 0FFFFH。然而在 78K0 微控制器中，从 0040H 到 007FH 的地址用于 CALLT 表，在 78K0R 微控制器中变为向量地址。

2.2.3 CALLT指令表区域

在 78K0R 微控制器中，从 0080H 到 00BFH 的 64 字节区域保留为 CALLT 指令表区域。

在 78K0 微控制器中使用单字节 CALL 指令，78K0R 微控制器使用 2 字节 CALL 指令。地址也已经相应改变。

由于地址码仅为 2 字节，中断转移目的起始地址为从 00000H 到 0FFFFH 的 64 KB 空间内。

2.3 内部数据存储器（内部RAM）空间

78K0 微控制器包括内部高速 RAM 和内部扩展 RAM，内部高速 RAM 可用作堆栈，内部扩展 RAM 可用于取指。而 78K0R 微控制器仅有 1 个 RAM 区域，可用于堆栈和取值。

地址范围的上限固定为 FFEFFH，根据产品 RAM 大小可以向下扩展。最大容量为 61.75 KB。地址范围的下限请参考各产品的用户手册。

在 78K0 和 78K0R 微控制器中 Saddr 空间和通用寄存器区域（从 FFEE0H 到 FFEFFH）具有相同地址。在内部 RAM 中设置堆栈区域。

2.4 特殊功能寄存器（SFR） 区域

SFR 具有特殊功能，与通用寄存器不同。

SFR 空间区域从 FFF00H 到 FFFFFH。

<R> SFRs 可以如通用寄存器一样使用运算，传输和位操作指令进行操作。根据 SFR 的类型，通过 1、8 和 16 位单元操作。

每个可操作位单元可以按如下指定：

- 1 位 操作
  描述由编译器保留的 1 位操作指令操作数的符号（sfr.bit）。这个操作也可以通过一个地址来指定。
- 8 位操作
  描述由编译器保留的 8 位操作指令操作数的符号（sfr）。这个操作也可以通过一个地址来指定。
- 16 位操作
  描述由编译器保留的 16 位操作指令操作数的符号（sfrp）。当指定一个地址时，要使用一个偶地址。

虽然 78K0R 微控制器的 SFR 的规范与 78K0 微控制器的相同，但是某些寄存器与 78K0 不同。详细情况请参考各产品的用户手册。

2.5 扩展SFR （第二SFR） 区域

扩展 SFR （第二 SFR）具有特殊功能，与通用寄存器不同。

扩展 SFR 空间从 F0000H 到 F07FFH。SFR 区域之外的 SFR（从 FFF00H 到 FFFFFH）分配到此扩展 SFR 空间。但是，扩展 SFR 区域访问长度比 SFR 区域多 1 个字节。

<R> 扩展 SFRs 可以如通用寄存器一样使用运算，传输和位操作指令进行操作。根据 SFR 的类型，通过 1、8 和 16 位单元操作。

每个可操作位单元可以按如下指定：

- 1 位操作
  描述由编译器保留的 1 位操作指令操作数的符号（!addr16.bit）。这个操作也可以通过一个地址来指定。
- 8 位操作
  描述由编译器保留的 8 位操作指令操作数的符号（!addr16）。这个操作也可以通过一个地址来指定。
- 16 位操作
  描述由编译器保留的 16 位操作指令操作数的符号（!addr16）。当指定一个地址时，要使用一个偶地址。

2.6 扩展存储器空间

通过设置存储器扩展模式寄存器可以访问外部存储器空间。范围是从 flash 存储器到 EDFFFH。

在分立模式中可用 28 个引脚（A19 到 A0 和 D7 到 D0）作为外部引脚。在复用模式中，可用 20 个引脚（A19 到 A8 和 AD7 到 AD0）。

当使用外部存储器时的引脚设置，请参考各产品的用户手册中描述端口功能的章节。

注意事项 当在外部存储器区域中读取指令时，通过在 flash 存储器或 RAM 存储器区域的转移指令（CALL 或 BR）开始执行，通过在外部存储器区域中的返回指令（RET，RETB 或 RETI）结束执行。
当 flash 存储器区域与一个外部存储器区域相邻时，程序不能顺序执行。

第三章　寄存器

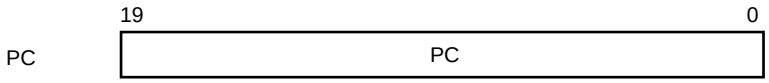
3.1　控制寄存器

控制寄存器具有特定的功能比如控制程序顺序，状态和堆栈存储器。控制寄存器包括程序计数器，程序状态字和堆栈指针。

3.1.1　程序计数器（PC）

程序计数器是 20 位的寄存器，可保持程序下次执行的地址信息。

图 3-1. 程序计数器的格式



3.1.2　程序状态字（PSW）

程序状态字（PSW）是一个 8 位寄存器，由各种标志位组成，通过指令执行对其进行置位或复位。

第 2 位加入 ISP1 标志，支持中断级别 4。

根据中断请求的产生或 PUSH PSW 指令执行，程序状态字的内容自动入栈；通过执行 RETB，RETI 和 POP PSW 指令，程序状态字的值自动恢复。

复位信号的产生将程序状态字的内容设置为 06H。

图 3-2. 程序状态字的格式

	<7>	<6>	<5>	<4>	<3>	<2>	<1>	0
PSW	IE	Z	RBS1	AC	RBS0	ISP1	ISP0	CY

（1） 中断允许标志（IE）

该标志用于控制 CPU 响应中断请求操作。

当 IE 为 0 时，表示不允许中断（DI），即禁止所有可屏蔽中断请求。

当 IE 为 1 时，表示允许中断（EI），用于各种中断源的中断屏蔽标志以及优先级规定标志来完成响应中断请求的控制。

当执行 DI 指令或中断请求得到响应时，该标志复位（0）；当执行 EI 指令时，该标志设置为 1。

（2）零标志（Z）

当操作结果为 0 时，该标志置 1。其他情况该标志置 0。

（3）寄存器 bank 选择标志（RBS0 和 RBS1）

2 位的标志，用来选择 4 个寄存器 bank 中的一个。

用来指明执行 SEL RBn 指令时所选择的寄存器组。

（4）半进位标志（AC）

如果操作结果中第 3 位有进位或在第 3 位上有借位，则该标志置 1。其他情况该标志置 0。

（5）优先服务标志（ISP0 和 ISP1）

该标志用来管理可屏蔽向量中断响应的优先级。由优先级指定标志寄存器（PR）指定的比 ISP0 和 ISP1 值低的向量中断请求被禁止响应。对请求的实际响应是由中断允许标志（IE）的状态控制的。

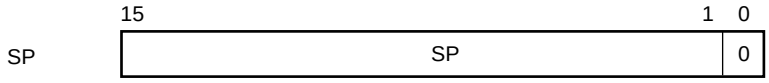
（6）进位标志（CY）

该标志存储的是在执行加减指令时出现的进位或借位。它也存储循环指令执行中的转移值，还可以在位操作指令执行中作为位累加器使用。

3.1.3 堆栈指针（SP）

这是一个 16 位的寄存器，用来存放存储器堆栈区的起始地址。内部 RAM 区域可设置为堆栈区。

图 3-3. 堆栈指针的格式



在向堆栈写（存）数据时，堆栈指针 SP 递减，而从堆栈中读出（恢复）数据时，堆栈指针累加。

由于 $\overline{\text{RESET}}$ 信号产生时，SP 的内容不确定，所以在使用堆栈前必须先对 SP 初始化。SP 地址必须是偶数。如果是奇数地址，则 LSB 固定为 0。

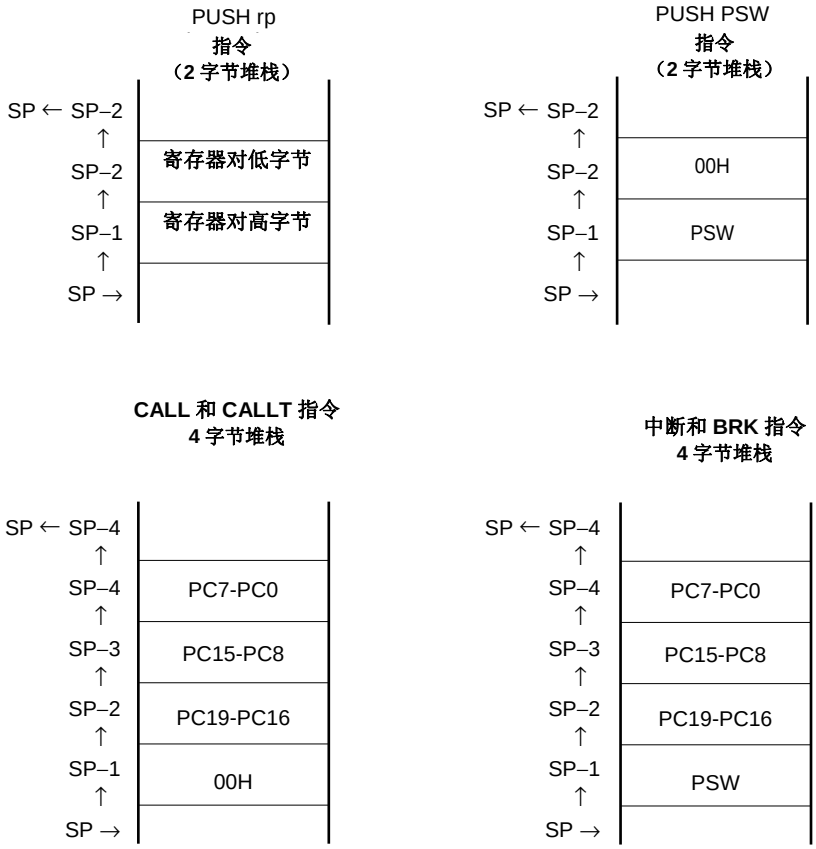
在 78K0R 微控制器中，由于扩展了存储器空间，用于 CALL 指令或中断的堆栈地址比 1 字节的范围大，由于用于堆栈的 RAM 为 16 位，因此使用的堆栈大小为 2 字节或 4 字节（参考 **表 3-1**）。

表 3-1. 78K0 微控制器与 78K0R 微控制器堆栈大小的区别

保存指令	恢复指令	78K0微控制器的堆栈大小	78K0R微控制器的堆栈大小
PUSH rp	POP rp	2 字节	2 字节
PUSH PSW	POP PSW	1 字节	2 字节
CALL， CALLT	RET	2 字节	4 字节
Interrupt	RETI	3 字节	4 字节
BRK	RETB	3 字节	4 字节

在 78K0R 微控制器中各种保存数据的堆栈操作如图 3-4 所示。

图 3-4. 将数据存入堆栈



只能在内部 RAM 中设置堆栈指针。目标地址范围从 F0000H 到 FFFFFH; 一定不要超过内部 RAM 空间。如果设置为内部 RAM 空间外的地址，则对此地址的写操作将被忽略，读操作将返回不确定的值。

3.2 通用寄存器

通用寄存器映射到 RAM 的地址空间为 FFEE0H 到 FFEFFH。通用寄存器共有四个 bank，每个 bank 由 8 个 8 位寄存器（X， A， C， B， E， D， L 和 H）组成。通过 CPU 控制指令“SEL RBn”可以设置指令执行所用的 bank。

此外每个寄存器可作为一个 8 位寄存器使用，两个成对的 8 位寄存器可作为一个 16 位寄存器使用。

编程时可以使用功能名称（X， A， C， B， E， D， L， H， AX， BC， DE 和 HL）和绝对名称（R0 到 R7，RP0 到 RP3）。

注意事项 禁止将通用寄存器空间（FFEE0H 到 FFEFFH）作为读取指令区域或堆栈区域使用。

表 3-2. 通用寄存器列表（适用于 78K0 微控制器）

Bank名称	寄存器				绝对地址
	功能名称		绝对名称		
	16位处理	8位处理	16位处理	8位处理	
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	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	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	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

3.3 ES和CS寄存器

78K0R 微控制器具有 ES 和 CS 寄存器。可以通过 ES 寄存器访问数据，通过 CS 寄存器指定转移指令的执行地址。关于如何使用这些寄存器，请参考 **第四章 寻址**。

复位后，ES 的初始值为 0FH，CS 的初始值为 00H。

图 3-5. ES 和 CS 寄存器的格式

	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	CP2	CP1	CP0

3.4 特殊功能寄存器（SFRs）

78K0R 微控制器中固定地址的 SFRs 如表 3-3 所示。

表 3-3. 固定地址的 SFRs 列表

地址	寄存器名称
FFFF8H	SPL
FFFF9H	SPH
FFFFAH	PSW
FFFFBH	保留
FFFFCH	CS
FFFFDH	ES
FFFFEH	PMC
FFFFFH	MEM

3.4.1 处理器模式控制寄存器（PMC）

处理器模式控制寄存器是一个 8 位寄存器，用于控制处理器模式。详细情况，请参考 **2.2 内部程序存储器空间。**

复位后 PMC 的初始值为 00H。

图 3-6. 处理器模式控制寄存器的格式

地址：FFFFEH 复位后：00H R/W

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

MAA	选择映射到从F0000H 到FFFFFH[注]区域的flash存储器空间
0	00000H 到 0FFFFH 映射到 F0000H 到 FFFFFH
1	10000H 到 1FFFFH 映射到 F0000H 到 FFFFFH

注 SFR 和 RAM 区域也在 F0000H 到 FFFFFH 范围中， 在重叠区域优先于其它区域。

注意事项 1. 初始化设置时仅设置 PMC 寄存器一次。除了初始化设置之外禁止重写 PMC。

2. 设置 PMC 后， 等待至少一个指令访问镜像区域。

第四章　寻址

寻址分为两类：数据地址的寻址和程序地址的寻址。每一种类型的寻址模式描述如下。

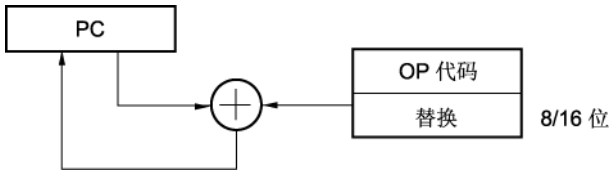
4.1　指令地址寻址

4.1.1　相对寻址

[功能]

相对寻址将指令字（有符号数的补码：–128 到+127 或–32768 到+32767）中的偏移量与程序计数器（PC）的值（下一条指令的起始地址）相加，结果赋给程序计数器（PC），然后转向相加结果指向的地址。相对寻址仅适用于转移指令。

图 4-1. 相对寻址图示



4.1.2 立即数寻址

[功能]

立即数寻址将指令中的立即数赋给程序计数器（PC），然后将程序地址作为分支地址。

对于立即数寻址，CALL !!addr20 或 BR !!addr20 用于指定 20 位地址，CALL !addr16 或 BR !addr16 用于指定 16 位地址。当指定 16 位地址时高 4 位设置为 0000。

图 4-2. CALL !!addr20/BR !!addr20 示例

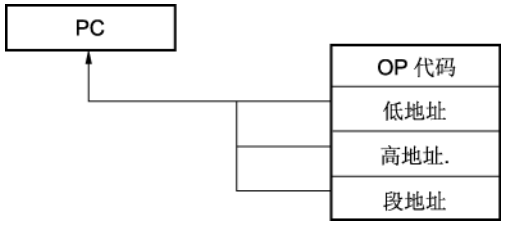
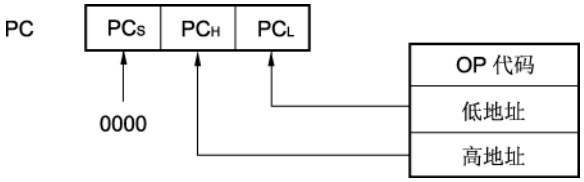


图 4-3. CALL !addr16/BR !addr16 示例



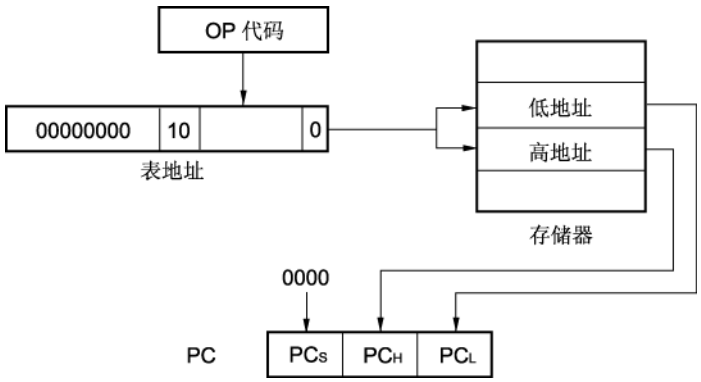
4.1.3 表间接寻址

[功能]

表间接寻址通过指令字中的 5 位立即数指定在 CALLT 表区域（0080H 到 00BFH）中的表地址，以 16 位数据存储在表地址和程序计数器（PC）的下个地址中，然后转向该地址执行程序。表间接寻址仅适用于 CALLT 指令。

在 78K0R 微控制器中，转移地址范围仅为 64 KB 空间，从 00000H 到 0FFFFH。

图 4-4. 表间接寻址图示

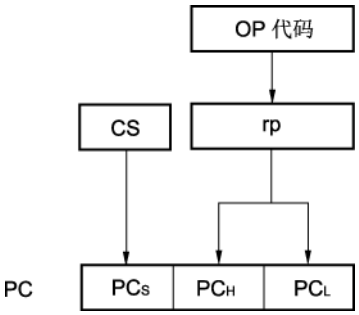


4.1.4 寄存器直接寻址

[功能]

寄存器直接寻址通过指令字的 20 位数据将当前寄存器 bank 的通用寄存器对（AX/BC/DE/HL）和 CS 寄存器的内容赋给程序计数器（PC），指定为程序地址。寄存器直接寻址仅适用于 CALL AX、BC、DE、HL 和 BR AX 指令。

图 4-5. 寄存器直接寻址图示



4.2 数据地址寻址

4.2.1 隐含寻址

[功能]

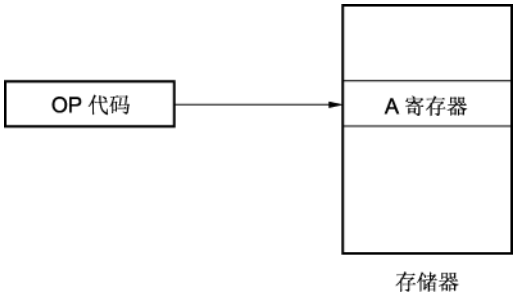
访问特殊功能寄存器（例如累加器）指令直接由指令字指定，无须使用指令字中任何寄存器指定域。

[操作数格式]

由于指令自动采用隐含寻址方式，所以没有特定的操作数格式。

隐含寻址仅适用于 MULU X。

图 4-6. 隐含寻址图示



4.2.2 寄存器寻址

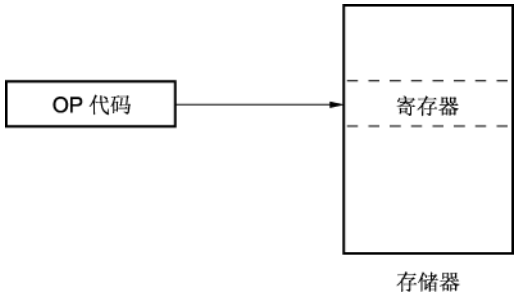
[功能]

寄存器寻址方式将通用寄存器作为操作数进行访问。3 位的指令字用于选择 8 位寄存器，2 位的指令字用于选择 16 位寄存器。

[操作数格式]

标识符	描述
r	X， A， C， B， E， D， L， H
rp	AX， BC， DE， HL

图 4-7. 寄存器寻址图示



4.2.3 直接寻址

[功能]

直接寻址将指令字中的立即数作为操作数地址直接指定目的地址。

[操作数格式]

标识符	描述
ADDR16	标号或16位立即数（仅为从F0000H 到 FFFFFH的空间）
ES:ADDR16	标号或16位立即数（通过ES寄存器指定高4位地址）

图 4-8. ADDR16 示例

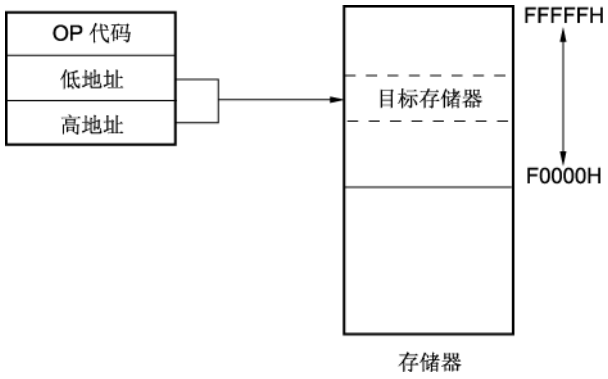
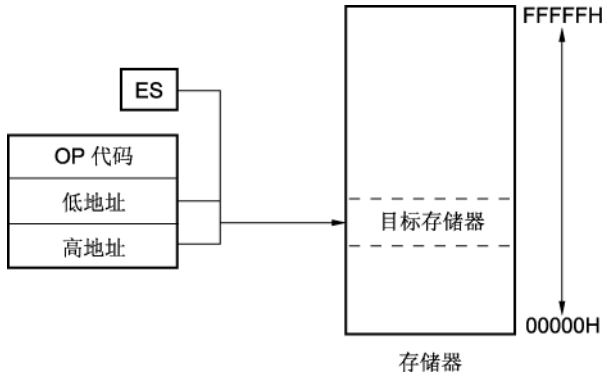


图 4-9. ES:ADDR16 示例



4.2.4 短直接寻址

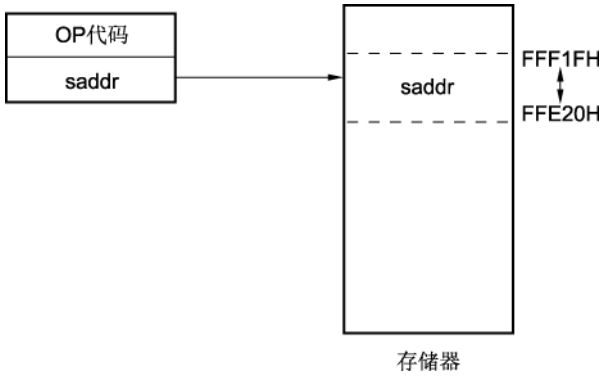
[功能]

短直接寻址用指令字中的 8 位数据直接指定目标地址。这种寻址仅适用于从 FFE20H 到 FFF1FH 的空间。

[操作数格式]

标识符	描述
SADDR	标号，FFE20H 到 FFF1FH 的立即数或 0FE20H 到 0FF1FH 的立即数 （仅从FFE20H 到FFF1FH的空间）
SADDRP	标号，FFE20H 到 FFF1FH的立即数或 0FE20H 到 0FF1FH的立即数（仅偶地址） （仅从FFE20H 到FFF1FH 的空间）

图 4-10. 短直接寻址图示



备注 SADDR 和 SADDRP 以 16 位立即数（实际地址的高 4 位忽略）表示地址 FE20H 到 FF1FH 的值，以 20 位立即数表示地址 FFE20H 到 FFF1FH 的值。

不管是否使用 SADDR 或 SADDRP，从 FFE20H 到 FFF1FH 空间中的地址指定为存储器。

4.2.5 SFR 寻址

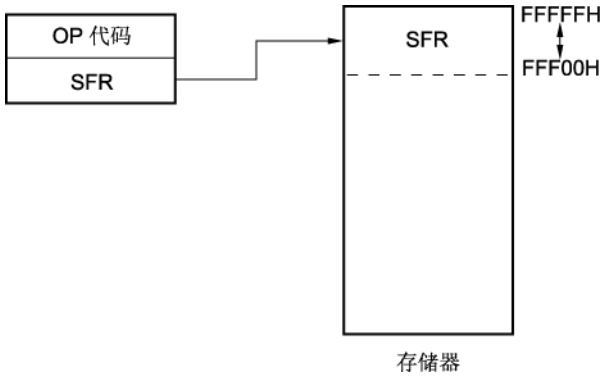
[功能]

SFR 寻址用指令字中的 8 位数据直接指定目标 SFR 地址。这种寻址仅适用于从 FFF00H 到 FFFFFH 的空间。

[操作数格式]

标识符	描述
SFR	SFR 名称
SFRP	16位操作SFR 名称（仅偶地址）

图 4-11. SFR 寻址图示



4.2.6 寄存器间接寻址

[功能]

寄存器间接寻址使用由指令字指定的寄存器对的内容将目的地址作为一个操作数地址。

[操作数格式]

标识符	描述
–	[DE], [HL] （仅为从F0000H 到 FFFFFH 的空间）
–	ES:[DE]， ES:[HL] （通过ES寄存器指定高4位地址）

图 4-12. [DE], [HL]示例

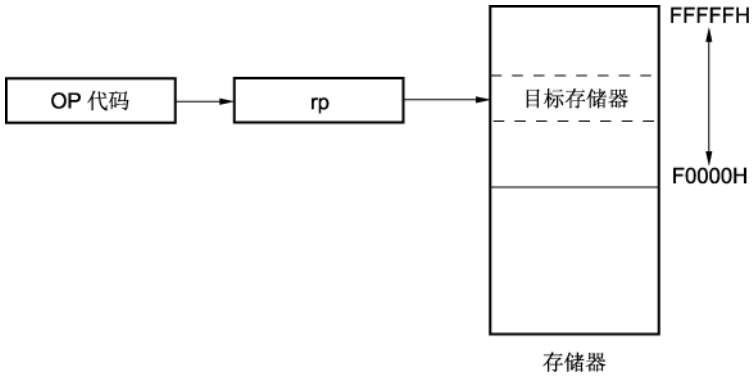
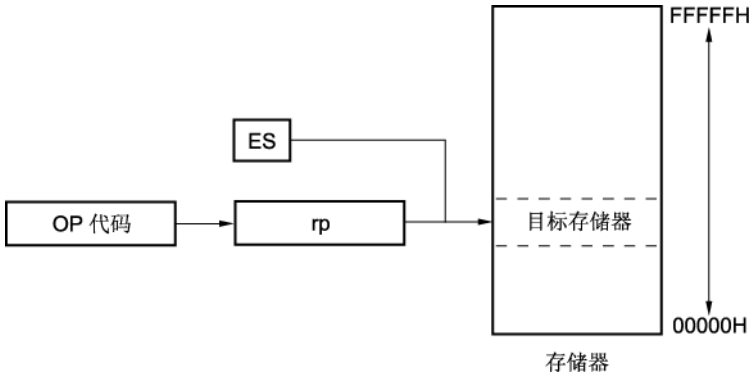


图 4-13. ES:[DE], ES:[HL]示例



4.2.7 基址寻址

[功能]

基址寻址由指令字指定寄存器对的值作为基址，8 位立即数或 16 位立即数作为偏移量。相加结果用于指定目标地址。

[操作数格式]

标识符	描述
–	[HL + byte]， [DE + byte]， [SP + byte] （仅为从F0000H 到 FFFFFH 的空间）
–	word[B]， word[C] （仅为从F0000H 到FFFFFH 的空间）
–	word[BC] （仅为从F0000H 到 FFFFFH 的空间）
–	ES:[HL + byte]， ES:[DE + byte] （通过ES寄存器指定高4位地址）
–	ES:word[B]， ES：word[C] （通过ES寄存器指定高4位地址）
–	ES:word[BC] （通过ES寄存器指定高4位地址）

图 4-14. [SP+byte]示例

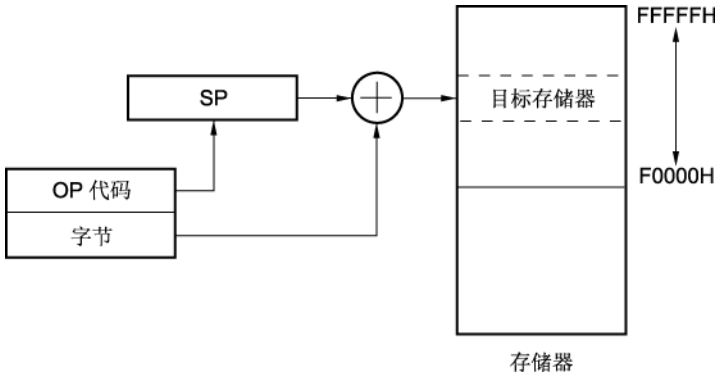


图 4-15. [HL + byte], [DE + byte]示例

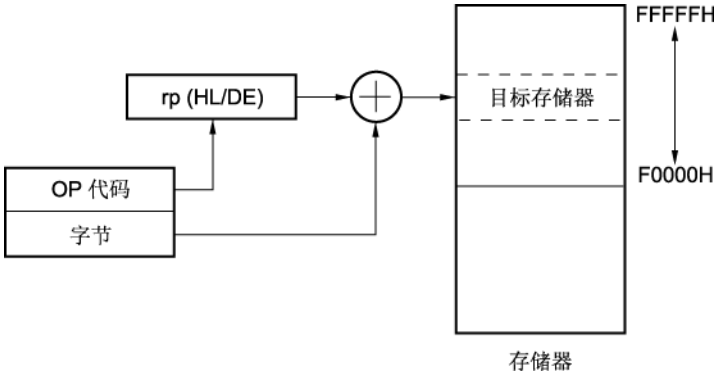


图 4-16. word[B], word[C]示例

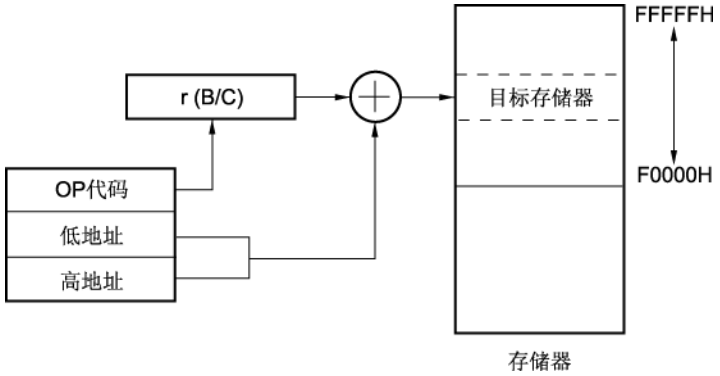


图 4-17. word[BC]示例

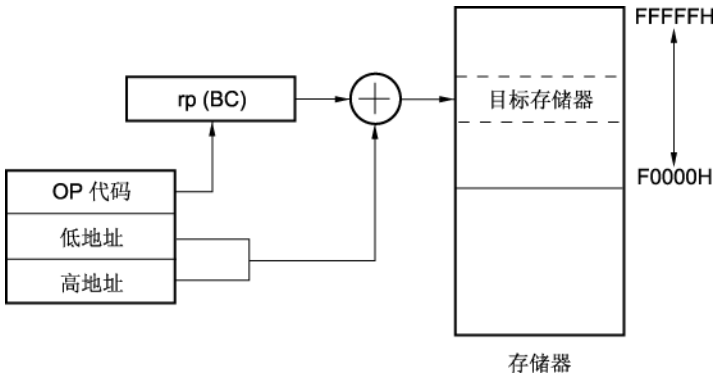


图 4-18. ES:[HL + byte]， ES:[DE + byte]示例

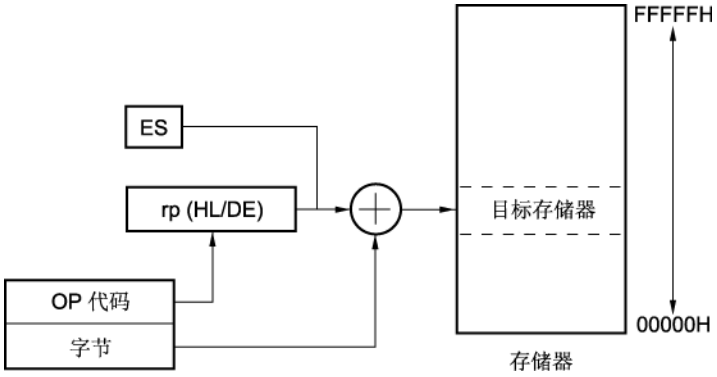


图 4-19. ES:word[B]， ES:word[C]示例

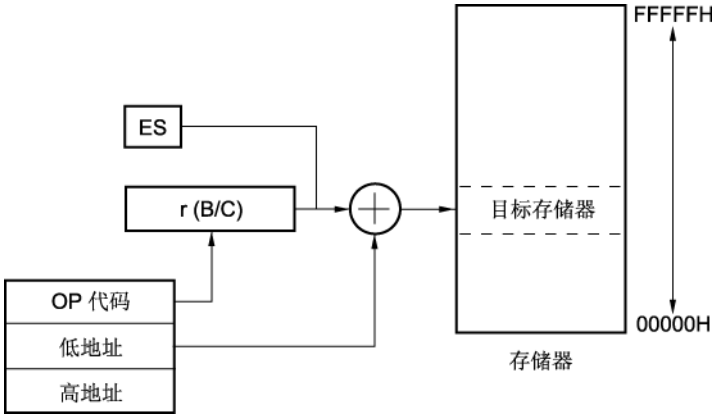
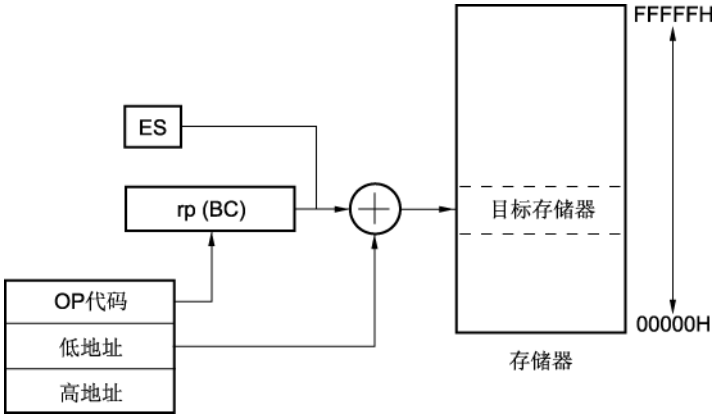


图 4-20. ES:word[BC]示例



4.2.8 基址变址寻址

[功能]

基址变址寻址由指令字指定寄存器对的值作为基址，将作为偏移量的 B 或 C 寄存器的内容加到 HL 寄存器中。相加结果用于指定目标地址。

[操作数格式]

标识符	描述
–	[HL+B]， [HL+C] （仅为从F0000H 到FFFFFH 的空间）
–	ES: [HL+B]， ES:[HL+C] （通过ES寄存器指定高4位地址）

图 4-21. [HL+B]， [HL+C]示例

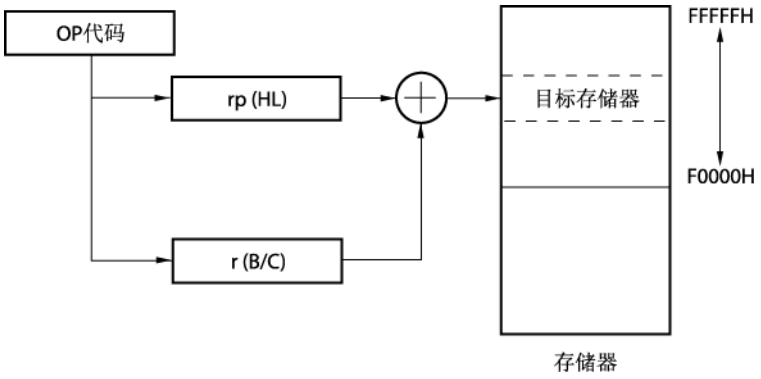
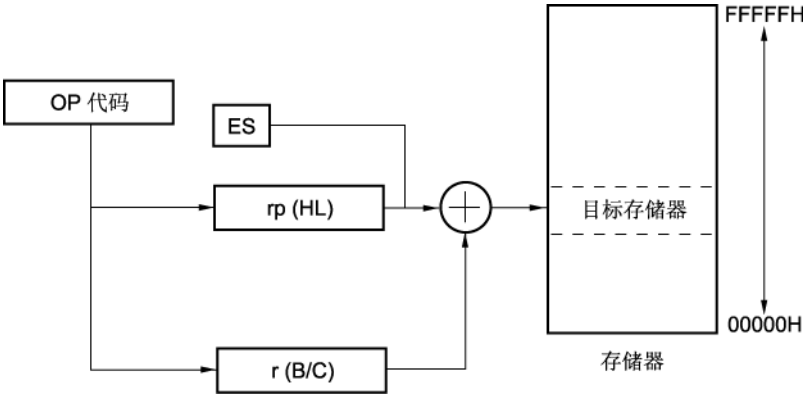


图 4-22. ES:[HL+B]， ES:[HL+C]示例



4.2.9 堆栈寻址

[功能]

根据堆栈指针（SP）的内容对堆栈区域进行间接寻址。当执行 PUSH、POP、子程序调用和返回指令时，或者产生中断请求时保存或恢复寄存器操作时，将自动采用这种寻址方式。

该方式仅对内部 RAM 区域进行寻址。

[操作数格式]

标识符	描述
–	PUSH AX/BC/DE/HL POP AX/BC/DE/HL CALL/CALLT RET BRK RETB （产生中断请求） RETI

第五章 指令集

本章列出了78K0R系列的指令集。适合所有的78K0R系列产品。

备注 在**5.5 操作列表**和 **5.6 指令格式列表**中的阴影部分表示78K0R 微控制器中新加的操作或指令格式。

5.1 操作数标识符和标识方法

根据规范确定的指令操作数标识方法（详情可参见汇编程序编程规范），在每种指令的操作数栏列出操作数。如果有两种或两种以上的标识方法，可选其中之一。大写字母和符号#、!、!!、$、$!、[ ]和ES:是关键字，必须按其原样书写。每种符号的含义如下所示。

- #： 立即数
- !： 16位绝对地址
- !!： 20位绝对地址
- $： 8位相对地址
- $!： 16位相对地址
- [ ]： 间接地址
- ES:：扩展地址

立即数用来描述一个数值型数据或标号。当使用标号时，注意必须加上#、!、!!、$、$!、[ ] 和ES:符号。

对应操作数寄存器标识符r和rp，功能名称 （X，A，C等）或绝对名称 （下表括号中的名称： R0，R1，R2等）都可用于标识。

表 5-1. 操作数标识符和标识方法

标识符	标识方法
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) 特殊功能寄存器符号（SFR 符号） 特殊功能寄存器符号（16位操作SFR符号。仅用于偶地址[注]）
saddr saddrp	FFE20H 到 FFF1FH立即数或标号 FFE20H 到 FF1FH 立即数或标号 （仅用于偶地址[注]）
addr20 addr16 addr5	00000H 到 FFFFFH 立即数或标号 0000H 到 FFFFH 立即数或标号 （仅用于16位数据传送指令的偶地址[注]） 0080H 到 00BFH 立即数或标号（仅用于偶地址）
word byte bit	16位立即数或标号 8位立即数或标号 3位立即数或标号
RBn	RB0到 RB3

注 当为奇地址时第0位 = 0 。

5.2 操作栏描述

执行指令时的操作在操作栏中如以下符号所示。

表 5-2. 操作栏中的符号

符号	功能
A	A 寄存器；8位累加器
X	X 寄存器
B	B 寄存器
C	C 寄存器
D	D 寄存器
E	E 寄存器
H	H 寄存器
L	L 寄存器
ES	ES 寄存器
CS	CS 寄存器
AX	AX 寄存器对；16位累加器
BC	BC 寄存器对
DE	DE 寄存器对
HL	HL 寄存器对
PC	程序计数器
SP	堆栈指针
PSW	程序状态字
CY	进位标志
AC	半进位标志
Z	零标志
RBS	寄存器bank 选择标志
IE	中断请求允许标志
()	括号中的地址或寄存器所指的存储单元的内容
X_H, X_L X_S, X_H, X_L	16位寄存器：X_H = 高8位，X_L = 低8位 20位寄存器:X_S = （第19 纟 16位），X_H = （第15 纟 8位），X_L = （第7 纟 0位）
∧	逻辑与（AND）
∨	逻辑或（OR）
⩛	逻辑异或（exclusive OR）
¯	数据取反
addr5	16位立即数（仅用于0080H到00BFH的偶地址）
addr16	16位立即数
addr20	20位立即数
jdisp8	有符号8位数据（偏移量）
jdisp16	有符号16位数据（偏移量）

5.3 标志栏的描述

执行指令时标志值的改变在标志栏中如以下符号所示。

表 5-3. 标志栏中的符号

符号	标志值的改变
(Blank)	不改变
0	清零
1	设置为1
×	根据结果设置/清零
R	恢复以前保存的值

5.4 PREFIX指令

带“ES:”的指令具有PREFIX操作码，通过将ES寄存器的值与F0000H到FFFFFH的64KB空间相加可以扩展数据访问区域到1MB空间。当PREFIX操作码作为PREFIX附加到目标指令时，执行PREFIX操作码之后仅一条指令的地址与ES寄存器的值相加。

表 5-4. PREFIX操作码使用示例

指令	操作码				
	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	–	–	–

注意事项 在执行PREFIX指令之前用MOV ES，A等设置ES寄存器的值。

5.5 操作列表

表 5-5. 操作列表（1/17）

指令组	助记符	操作数	字节	时钟		操作	标志
				注 1	注 2		Z AC CY
8位数据传送	MOV	r, #byte	2	1	–	r ← byte	
		saddr, #byte	3	1	–	（saddr） ← byte	
		sfr, #byte	3	1	–	sfr ← byte	
		!addr16, #byte	4	1	–	（addr16） ← byte	
		A, r 注 3	1	1	–	A ← r	
		r, A 注 3	1	1	–	r ← A	
		A, saddr	2	1	–	A ← （saddr）	
		saddr, A	2	1	–	（saddr） ← A	
		A, sfr	2	1	–	A ← sfr	
		sfr, A	2	1	–	sfr ← A	
		A, !addr16	3	1	4	A ← （addr16）	
		!addr16, A	3	1	–	（addr16） ← A	
		PSW, #byte	3	3	–	PSW ← byte	× × ×
		A, PSW	2	1	–	A ← PSW	
		PSW, A	2	3	–	PSW ← A	× × ×
		ES, #byte	2	1	–	ES ← byte	
		ES, saddr	3	1	–	ES ← （saddr）	
		A, ES	2	1	–	A ← ES	
		ES, A	2	1	–	ES ← A	
		CS, #byte	3	1	–	CS ← byte	
		A, CS	2	1	–	A ← CS	
		CS, A	2	1	–	CS ← A	
		A, [DE]	1	1	4	A ← （DE）	
		[DE], A	1	1	–	（DE） ← A	
		[DE + byte], #byte	3	1	–	（DE + byte） ← byte	
		A, [DE + byte]	2	1	4	A ← （DE + byte）	
		[DE + byte], A	2	1	–	（DE + byte） ← A	
		A, [HL]	1	1	4	A ← （HL）	
		[HL], A	1	1	–	（HL） ← A	
		[HL + byte], #byte	3	1	–	（HL + byte） ← byte	

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（2/17）

指令组	助记符	操作数	字节	时钟		操作	标志
				注 1	注 2		Z AC CY
8位数据传送	MOV	A, [HL + byte]	2	1	4	A ← （HL + byte）	
		[HL + byte], A	2	1	–	（HL + byte） ← A	
		A, [HL + B]	2	1	4	A ← （HL + B）	
		[HL + B], A	2	1	–	（HL + B） ← A	
		A, [HL + C]	2	1	4	A ← （HL + C）	
		[HL + C], A	2	1	–	（HL + C） ← A	
		word[B], #byte	4	1	–	（B + word） ← byte	
		A, word[B]	3	1	4	A ← （B + word）	
		word[B], A	3	1	–	（B + word） ← A	
		word[C], #byte	4	1	–	（C + word） ← byte	
		A, word[C]	3	1	4	A ← （C + word）	
		word[C], A	3	1	–	（C + word） ← A	
		word[BC], #byte	4	1	–	（BC + word） ← byte	
		A, word[BC]	3	1	4	A ← （BC + word）	
		word[BC], A	3	1	–	（BC + word） ← A	
		[SP + byte], #byte	3	1	–	（SP + byte） ← byte	
		A, [SP + byte]	2	1	–	A ← （SP + byte）	
		[SP + byte], A	2	1	–	（SP + byte） ← A	
		B, saddr	2	1	–	B ← （saddr）	
		B, !addr16	3	1	4	B ← （addr16）	
		C, saddr	2	1	–	C ← （saddr）	
		C, !addr16	3	1	4	C ← （addr16）	
		X, saddr	2	1	–	X ← （saddr）	
		X, !addr16	3	1	4	X ← （addr16）	
		ES:!addr16, #byte	5	2	–	（ES, addr16） ← byte	
		A, ES:!addr16	4	2	5	A ← （ES, addr16）	
		ES:!addr16, A	4	2	–	（ES, addr16） ← A	
		A, ES:[DE]	2	2	5	A ← （ES, DE）	
		ES:[DE], A	2	2	–	（ES, DE） ← A	
		ES:[DE + byte],#byte	4	2	–	（（ES, DE） + byte） ← byte	
		A, ES:[DE + byte]	3	2	5	A ← （（ES, DE） + byte）	
		ES:[DE + byte], A	3	2	–	（（ES, DE） + byte） ← A	

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。

2. 当访问程序存储器区域时。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。

2. 该时钟周期用于内部ROM（flash存储器）程序。

3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（3/17）

指令组	助记符	操作数	字节	时钟		操作	标志
				注 1	注 2		Z AC CY
8位数据传送	MOV	A, ES:[HL]	2	2	5	A ← （ES, HL）	
		ES:[HL], A	2	2	–	（ES, HL） ← A	
		ES:[HL + byte],#byte	4	2	–	（（ES, HL） + byte） ← byte	
		A, ES:[HL + byte]	3	2	5	A ← （（ES, HL） + byte）	
		ES:[HL + byte], A	3	2	–	（（ES, HL） + byte） ← A	
		A, ES:[HL + B]	3	2	5	A ← （（ES, HL） + B）	
		ES:[HL + B], A	3	2	–	（（ES, HL） + B） ← A	
		A, ES:[HL + C]	3	2	5	A ← （（ES, HL） + C）	
		ES:[HL + C], A	3	2	–	（（ES, HL） + C） ← A	
		ES:word[B], #byte	5	2	–	（（ES, B） + word） ← byte	
		A, ES:word[B]	4	2	5	A ← （（ES, B） + word）	
		ES:word[B], A	4	2	–	（（ES, B） + word） ← A	
		ES:word[C], #byte	5	2	–	（（ES, C） + word） ← byte	
		A, ES:word[C]	4	2	5	A ← （（ES, C） + word）	
		ES:word[C], A	4	2	–	（（ES, C） + word） ← A	
		ES:word[BC], #byte	5	2	–	（（ES, BC） + word） ← byte	
		A, ES:word[BC]	4	2	5	A ← （（ES, BC） + word）	
		ES:word[BC], A	4	2	–	（（ES, BC） + word） ← A	
		B, ES:!addr16	4	2	5	B ← （ES, addr16）	
		C, ES:!addr16	4	2	5	C ← （ES, addr16）	
		X, ES:!addr16	4	2	5	X ← （ES, addr16）	
	XCH	A, r 注 3	1（r = X） 2（r = X 除外）	1	–	A ← → r	
		A, saddr	3	2	–	A ← → （saddr）	
		A, sfr	3	2	–	A ← → sfr	
		A, !addr16	4	2	–	A ← → （addr16）	
		A, [DE]	2	2	–	A ← → （DE）	
		A, [DE + byte]	3	2	–	A ← → （DE + byte）	
		A, [HL]	2	2	–	A ← → （HL）	
		A, [HL + byte]	3	2	–	A ← → （HL + byte）	
		A, [HL + B]	2	2	–	A ← → （HL + B）	
		A, [HL + C]	2	2	–	A ← → （HL + C）	

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（4/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
8位数据传送	XCH	A, ES:!addr16	5	3	–	A ← → （ES, addr16）			
		A, ES:[DE]	3	3	–	A ← → （ES, DE）			
		A, ES:[DE + byte]	4	3	–	A ← → （（ES, DE） + byte）			
		A, ES:[HL]	3	3	–	A ← → （ES, HL）			
		A, ES:[HL + byte]	4	3	–	A ← → （（ES, HL） + byte）			
		A, ES:[HL + B]	3	3	–	A ← → （（ES, HL） + B）			
		A, ES:[HL + C]	3	3	–	A ← → （（ES, HL） + C）			
	ONEB	A	1	1	–	A ← 01H			
		X	1	1	–	X ← 01H			
		B	1	1	–	B ← 01H			
		C	1	1	–	C ← 01H			
		saddr	2	1	–	（saddr） ← 01H			
		!addr16	3	1	–	（addr16） ← 01H			
		ES:!addr16	4	2	–	（ES, addr16） ← 01H			
	CLRB	A	1	1	–	A ← 00H			
		X	1	1	–	X ← 00H			
		B	1	1	–	B ← 00H			
		C	1	1	–	C ← 00H			
		saddr	2	1	–	（saddr） ← 00H			
		!addr16	3	1	–	（addr16） ← 00H			
		ES:!addr16	4	2	–	（ES,addr16） ← 00H			
	MOVS	[HL + byte], X	3	1	–	（HL + byte） ← X	×		×
		ES:[HL + byte], X	4	2	–	（ES, HL + byte） ← X	×		×
16位数据传送	MOVW	rp, #word	3	1	–	rp ← word			
		saddrp, #word	4	1	–	（saddrp） ← word			
		sfrp, #word	4	1	–	sfrp ← word			
		AX, saddrp	2	1	–	AX ← （saddrp）			
		saddrp, AX	2	1	–	（saddrp） ← AX			
		AX, sfrp	2	1	–	AX ← sfrp			
		sfrp, AX	2	1	–	sfrp ← AX			
		AX, rp 注 3	1	1	–	AX ← rp			
		rp, AX 注 3	1	1	–	rp ← AX			

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. rp = AX 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（5/17）

指令组	助记符	操作数	字节	时钟		操作	标志
				注 1	注 2		Z AC CY
16位数据传送	MOVW	AX, !addr16	3	1	4	AX ← （addr16）	
		!addr16, AX	3	1	–	（addr16） ← AX	
		AX, [DE]	1	1	4	AX ← （DE）	
		[DE], AX	1	1	–	（DE） ← AX	
		AX, [DE + byte]	2	1	4	AX ← （DE + byte）	
		[DE + byte], AX	2	1	–	（DE + byte） ← AX	
		AX, [HL]	1	1	4	AX ← （HL）	
		[HL], AX	1	1	–	（HL） ← AX	
		AX, [HL + byte]	2	1	4	AX ← （HL + byte）	
		[HL + byte], AX	2	1	–	（HL + byte） ← AX	
		AX, word[B]	3	1	4	AX ← （B + word）	
		word[B], AX	3	1	–	（B + word） ← AX	
		AX, word[C]	3	1	4	AX ← （C + word）	
		word[C], AX	3	1	–	（C + word） ← AX	
		AX, word[BC]	3	1	4	AX ← （BC + word）	
		word[BC], AX	3	1	–	（BC + word） ← AX	
		AX, [SP + byte]	2	1	–	AX ← （SP + byte）	
		[SP + byte], AX	2	1	–	（SP + byte） ← AX	
		BC, saddrp	2	1	–	BC ← （saddrp）	
		BC, !addr16	3	1	4	BC ← （addr16）	
		DE, saddrp	2	1	–	DE ← （saddrp）	
		DE, !addr16	3	1	4	DE ← （addr16）	
		HL, saddrp	2	1	–	HL ← （saddrp）	
		HL, !addr16	3	1	4	HL ← （addr16）	
		AX, ES:!addr16	4	2	5	AX ← （ES, addr16）	
		ES:!addr16, AX	4	2	–	（ES, addr16） ← AX	
		AX, ES:[DE]	2	2	5	AX ← （ES, DE）	
		ES:[DE], AX	2	2	–	（ES, DE） ← AX	
		AX, ES:[DE + byte]	3	2	5	AX ← （（ES, DE） + byte）	
		ES:[DE + byte], AX	3	2	–	（（ES, DE） + byte） ← AX	
		AX, ES:[HL]	2	2	5	AX ← （ES, HL）	
		ES:[HL], AX	2	2	–	（ES, HL） ← AX	

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（6/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
16位数据传送	MOVW	AX, ES:[HL + byte]	3	2	5	AX ← （（ES, HL） + byte）			
		ES:[HL + byte], AX	3	2	–	（（ES, HL） + byte） ← AX			
		AX, ES:word[B]	4	2	5	AX ← （（ES, B） + word）			
		ES:word[B], AX	4	2	–	（（ES, B） + word） ← AX			
		AX, ES:word[C]	4	2	5	AX ← （（ES, C） + word）			
		ES:word[C], AX	4	2	–	（（ES, C） + word） ← AX			
		AX, ES:word[BC]	4	2	5	AX ← （（ES, BC） + word）			
		ES:word[BC], AX	4	2	–	（（ES, BC） + word） ← AX			
		BC, ES:!addr16	4	2	5	BC ← （ES, addr16）			
		DE, ES:!addr16	4	2	5	DE ← （ES, addr16）			
		HL, ES:!addr16	4	2	5	HL ← （ES, addr16）			
	XCHW	AX, rp 注 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位操作	ADD	A, #byte	2	1	–	A, CY ← A + byte	×	×	×
		saddr, #byte	3	2	–	（saddr）, CY ← （saddr） + byte	×	×	×
		A, r 注 4	2	1	–	A, CY ← A + r	×	×	×
		r, A	2	1	–	r, CY ← r + A	×	×	×
		A, saddr	2	1	–	A, CY ← A + （saddr）	×	×	×
		A, !addr16	3	1	4	A, CY ← A + （addr16）	×	×	×
		A, [HL]	1	1	4	A, CY ← A + （HL）	×	×	×
		A, [HL + byte]	2	1	4	A, CY ← A + （HL + byte）	×	×	×
		A, [HL + B]	2	1	4	A, CY ← A + （HL + B）	×	×	×
		A, [HL + C]	2	1	4	A, CY ← A + （HL + C）	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A + （ES, addr16）	×	×	×
		A, ES:[HL]	2	2	5	A,CY ← A + （ES, HL）	×	×	×
		A, ES:[HL + byte]	3	2	5	A,CY ← A + （（ES, HL） + byte）	×	×	×
		A, ES:[HL + B]	3	2	5	A,CY ← A + （（ES, HL） + B）	×	×	×
		A, ES:[HL + C]	3	2	5	A,CY ← A + （（ES, HL） + C）	×	×	×

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. rp = AX 除外
4. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（7/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
8位操作	ADDC	A, #byte	2	1	–	A, CY ← A + byte + CY	×	×	×
		saddr, #byte	3	2	–	（saddr）, CY ← （saddr） + byte + CY	×	×	×
		A, r 注 3	2	1	–	A, CY ← A + r + CY	×	×	×
		r, A	2	1	–	r, CY ← r + A + CY	×	×	×
		A, saddr	2	1	–	A, CY ← A + （saddr） + CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A + （addr16） + CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A + （HL） + CY	×	×	×
		A, [HL + byte]	2	1	4	A, CY ← A + （HL + byte） + CY	×	×	×
		A, [HL + B]	2	1	4	A, CY ← A + （HL + B） + CY	×	×	×
		A, [HL + C]	2	1	4	A, CY ← A + （HL + C） + CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A + （ES, addr16） + CY	×	×	×
		A, ES:[HL]	2	2	5	A, CY ← A + （ES, HL） + CY	×	×	×
		A, ES:[HL + byte]	3	2	5	A, CY ← A + （（ES, HL） + byte） + CY	×	×	×
		A, ES:[HL + B]	3	2	5	A, CY ← A + （（ES, HL） + B） + CY	×	×	×
		A, ES:[HL + C]	3	2	5	A, CY ← A + （（ES, HL） + C） + CY	×	×	×
	SUB	A, #byte	2	1	–	A, CY ← A – byte	×	×	×
		saddr, #byte	3	2	–	（saddr）, CY ← （saddr） – byte	×	×	×
		A, r 注 3	2	1	–	A, CY ← A – r	×	×	×
		r, A	2	1	–	r, CY ← r – A	×	×	×
		A, saddr	2	1	–	A, CY ← A –（saddr）	×	×	×
		A, !addr16	3	1	4	A, CY ← A –（addr16）	×	×	×
		A, [HL]	1	1	4	A, CY ← A –（HL）	×	×	×
		A, [HL + byte]	2	1	4	A, CY ← A –（HL + byte）	×	×	×
		A, [HL + B]	2	1	4	A, CY ← A –（HL + B）	×	×	×
		A, [HL + C]	2	1	4	A, CY ← A –（HL + C）	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A –（ES：addr16）	×	×	×
		A, ES:[HL]	2	2	5	A, CY ← A –（ES：HL）	×	×	×
		A, ES:[HL + byte]	3	2	5	A, CY ← A –（（ES：HL） + byte）	×	×	×
		A, ES:[HL + B]	3	2	5	A, CY ← A –（（ES：HL） + B）	×	×	×
		A, ES:[HL + C]	3	2	5	A, CY ← A –（（ES：HL） + C）	×	×	×

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（8/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
8位操作	SUBC	A, #byte	2	1	–	A, CY ← A – byte – CY	×	×	×
		saddr, #byte	3	2	–	（saddr）, CY ← （saddr） – byte – CY	×	×	×
		A, r 注 3	2	1	–	A, CY ← A – r – CY	×	×	×
		r, A	2	1	–	r, CY ← r – A – CY	×	×	×
		A, saddr	2	1	–	A, CY ← A –（saddr） – CY	×	×	×
		A, !addr16	3	1	4	A, CY ← A –（addr16） – CY	×	×	×
		A, [HL]	1	1	4	A, CY ← A –（HL） – CY	×	×	×
		A, [HL + byte]	2	1	4	A, CY ← A –（HL + byte） – CY	×	×	×
		A, [HL + B]	2	1	4	A, CY ← A –（HL + B） – CY	×	×	×
		A, [HL + C]	2	1	4	A, CY ← A –（HL + C） – CY	×	×	×
		A, ES:!addr16	4	2	5	A, CY ← A –（ES：addr16） – CY	×	×	×
		A, ES:[HL]	2	2	5	A, CY ← A –（ES：HL） – CY	×	×	×
		A, ES:[HL + byte]	3	2	5	A, CY ← A –（（ES：HL） + byte） – CY	×	×	×
		A, ES:[HL + B]	3	2	5	A, CY ← A –（（ES：HL） + B） – CY	×	×	×
		A, ES:[HL + C]	3	2	5	A, CY ← A –（（ES：HL） + C） – CY	×	×	×
	AND	A, #byte	2	1	–	A ← A ∧ byte	×		
		saddr, #byte	3	2	–	（saddr） ←（saddr） ∧ byte	×		
		A, r 注 3	2	1	–	A ← A ∧ r	×		
		r, A	2	1	–	r ← r ∧ A	×		
		A, saddr	2	1	–	A ← A ∧（saddr）	×		
		A, !addr16	3	1	4	A ← A ∧（addr16）	×		
		A, [HL]	1	1	4	A ← A ∧（HL）	×		
		A, [HL + byte]	2	1	4	A ← A ∧（HL + byte）	×		
		A, [HL + B]	2	1	4	A ← A ∧（HL + B）	×		
		A, [HL + C]	2	1	4	A ← A ∧（HL + C）	×		
		A, ES:!addr16	4	2	5	A ← A ∧（ES：addr16）	×		
		A, ES:[HL]	2	2	5	A ← A ∧（ES：HL）	×		
		A, ES:[HL + byte]	3	2	5	A ← A ∧（（ES：HL） + byte）	×		
		A, ES:[HL + B]	3	2	5	A ← A ∧（（ES：HL） + B）	×		
		A, ES:[HL + C]	3	2	5	A ← A ∧（（ES：HL） + C）	×		

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（9/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
8位操作	OR	A, #byte	2	1	–	A ← A ∨ byte	×		
		saddr, #byte	3	2	–	（saddr） ← （saddr） ∨ byte	×		
		A, r 注 3	2	1	–	A ← A ∨ r	×		
		r, A	2	1	–	r ← r ∨ A	×		
		A, saddr	2	1	–	A ← A ∨ （saddr）	×		
		A, !addr16	3	1	4	A ← A ∨ （addr16）	×		
		A, [HL]	1	1	4	A ← A ∨ （HL）	×		
		A, [HL + byte]	2	1	4	A ← A ∨ （HL + byte）	×		
		A, [HL + B]	2	1	4	A ← A ∨ （HL + B）	×		
		A, [HL + C]	2	1	4	A ← A ∨ （HL + C）	×		
		A, ES:!addr16	4	2	5	A ← A ∨ （ES：addr16）	×		
		A, ES:[HL]	2	2	5	A ← A ∨ （ES：HL）	×		
		A, ES:[HL + byte]	3	2	5	A ← A ∨ （（ES：HL） + byte）	×		
		A, ES:[HL + B]	3	2	5	A ← A ∨ （（ES：HL） + B）	×		
		A, ES:[HL + C]	3	2	5	A ← A ∨ （（ES：HL） + C）	×		
	XOR	A, #byte	2	1	–	A ← A ⩒ byte	×		
		saddr, #byte	3	2	–	（saddr） ← （saddr） ⩒ byte	×		
		A, r 注 3	2	1	–	A ← A ⩒ r	×		
		r, A	2	1	–	r ← r ⩒ A	×		
		A, saddr	2	1	–	A ← A ⩒ （saddr）	×		
		A, !addr16	3	1	4	A ← A ⩒ （addr16）	×		
		A, [HL]	1	1	4	A ← A ⩒ （HL）	×		
		A, [HL + byte]	2	1	4	A ← A ⩒ （HL + byte）	×		
		A, [HL + B]	2	1	4	A ← A ⩒ （HL + B）	×		
		A, [HL + C]	2	1	4	A ← A ⩒ （HL + C）	×		
		A, ES:!addr16	4	2	5	A ← A ⩒ （ES：addr16）	×		
		A, ES:[HL]	2	2	5	A ← A ⩒ （ES：HL）	×		
		A, ES:[HL + byte]	3	2	5	A ← A ⩒ （（ES：HL） + byte）	×		
		A, ES:[HL + B]	3	2	5	A ← A ⩒ （（ES：HL） + B）	×		
		A, ES:[HL + C]	3	2	5	A ← A ⩒ （（ES：HL） + C）	×		

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（10/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
8位操作	CMP	A, #byte	2	1	–	A – byte	×	×	×
		saddr, #byte	3	1	–	（saddr） – byte	×	×	×
		A, r 注 3	2	1	–	A – r	×	×	×
		r, A	2	1	–	r – A	×	×	×
		A, saddr	2	1	–	A –（saddr）	×	×	×
		A, !addr16	3	1	4	A –（addr16）	×	×	×
		A, [HL]	1	1	4	A –（HL）	×	×	×
		A, [HL + byte]	2	1	4	A –（HL + byte）	×	×	×
		A, [HL + B]	2	1	4	A –（HL + B）	×	×	×
		A, [HL + C]	2	1	4	A –（HL + C）	×	×	×
		!addr16, #byte	4	1	4	（addr16） – byte	×	×	×
		A, ES:!addr16	4	2	5	A –（ES：addr16）	×	×	×
		A, ES:[HL]	2	2	5	A –（ES：HL）	×	×	×
		A, ES:[HL + byte]	3	2	5	A –（（ES：HL） + byte）	×	×	×
		A, ES:[HL + B]	3	2	5	A –（（ES：HL） + B）	×	×	×
		A, ES:[HL + C]	3	2	5	A –（（ES：HL） + C）	×	×	×
		ES:!addr16, #byte	5	2	5	（ES：addr16） – byte	×	×	×
	CMP0	A	1	1	–	A – 00H	×	×	×
		X	1	1	–	X – 00H	×	×	×
		B	1	1	–	B – 00H	×	×	×
		C	1	1	–	C – 00H	×	×	×
		saddr	2	1	–	（saddr） – 00H	×	×	×
		!addr16	3	1	4	（addr16） – 00H	×	×	×
		ES:!addr16	4	2	5	（ES：addr16） – 00H	×	×	×
	CMPS	X, [HL + byte]	3	1	4	X –（HL + byte）	×	×	×
		X, ES:[HL + byte]	4	2	5	X –（（ES：HL） + byte）	×	×	×

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. r = A 除外

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（11/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
16位操作	ADDW	AX, #word	3	1	–	AX, CY ← AX + word	×	×	×
		AX, AX	1	1	–	AX, CY ← AX + AX	×	×	×
		AX, BC	1	1	–	AX, CY ← AX + BC	×	×	×
		AX, DE	1	1	–	AX, CY ← AX + DE	×	×	×
		AX, HL	1	1	–	AX, CY ← AX + HL	×	×	×
		AX, saddrp	2	1	–	AX, CY ← AX + （saddrp）	×	×	×
		AX, !addr16	3	1	4	AX, CY ← AX + （addr16）	×	×	×
		AX, [HL+byte]	3	1	4	AX, CY ← AX + （HL + byte）	×	×	×
		AX, ES:!addr16	4	2	5	AX, CY ← AX + （ES：addr16）	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX, CY ← AX + （（ES：HL） + byte）	×	×	×
	SUBW	AX, #word	3	1	–	AX, CY ← AX – word	×	×	×
		AX, BC	1	1	–	AX, CY ← AX – BC	×	×	×
		AX, DE	1	1	–	AX, CY ← AX – DE	×	×	×
		AX, HL	1	1	–	AX, CY ← AX – HL	×	×	×
		AX, saddrp	2	1	–	AX, CY ← AX – （saddrp）	×	×	×
		AX, !addr16	3	1	4	AX, CY ← AX – （addr16）	×	×	×
		AX, [HL+byte]	3	1	4	AX, CY ← AX – （HL + byte）	×	×	×
		AX, ES:!addr16	4	2	5	AX, CY ← AX – （ES：addr16）	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX, CY ← AX – （（ES：HL） + byte）	×	×	×
	CMPW	AX, #word	3	1	–	AX – word	×	×	×
		AX, BC	1	1	–	AX – BC	×	×	×
		AX, DE	1	1	–	AX – DE	×	×	×
		AX, HL	1	1	–	AX – HL	×	×	×
		AX, saddrp	2	1	–	AX – （saddrp）	×	×	×
		AX, !addr16	3	1	4	AX – （addr16）	×	×	×
		AX, [HL+byte]	3	1	4	AX – （HL + byte）	×	×	×
		AX, ES:!addr16	4	2	5	AX – （ES：addr16）	×	×	×
		AX, ES: [HL+byte]	4	2	5	AX – （（ES：HL） + byte）	×	×	×
乘法	MULU	X	1	1	–	AX ← A × X			

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（12/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
加/减	INC	r	1	1	–	r ← r + 1	×	×	
		saddr	2	2	–	（saddr） ← （saddr） + 1	×	×	
		!addr16	3	2	–	（addr16） ← （addr16） + 1	×	×	
		[HL+byte]	3	2	–	（HL+byte） ← （HL+byte） + 1	×	×	
		ES:!addr16	4	3	–	（ES, addr16） ← （ES, addr16） + 1	×	×	
		ES: [HL+byte]	4	3	–	((ES：HL)+byte) ← ((ES：HL) + byte) + 1	×	×	
	DEC	r	1	1	–	r ← r – 1	×	×	
		saddr	2	2	–	（saddr） ← （saddr） – 1	×	×	
		!addr16	3	2	–	（addr16） ← （addr16） – 1	×	×	
		[HL+byte]	3	2	–	（HL+byte） ← （HL+byte） – 1	×	×	
		ES:!addr16	4	3	–	（ES, addr16） ← （ES, addr16） – 1	×	×	
		ES: [HL+byte]	4	3	–	((ES：HL)+byte) ← ((ES：HL) + byte) – 1	×	×	
	INCW	rp	1	1	–	rp ← rp + 1			
		saddrp	2	2	–	（saddrp） ← （saddrp） + 1			
		!addr16	3	2	–	（addr16） ← （addr16） + 1			
		[HL+byte]	3	2	–	（HL+byte） ← （HL+byte） + 1			
		ES:!addr16	4	3	–	（ES, addr16） ← （ES, addr16） + 1			
		ES: [HL+byte]	4	3	–	((ES：HL)+byte) ← ((ES：HL) + byte) + 1			
	DECW	rp	1	1	–	rp ← rp – 1			
		saddrp	2	2	–	（saddrp） ← （saddrp） – 1			
		!addr16	3	2	–	（addr16） ← （addr16） – 1			
		[HL+byte]	3	2	–	（HL+byte） ← （HL+byte） – 1			
		ES:!addr16	4	3	–	（ES, addr16） ← （ES, addr16） – 1			
		ES: [HL+byte]	4	3	–	((ES：HL)+byte) ← ((ES：HL) + byte) – 1			
移位	SHR	A, cnt	2	1	–	（CY ← A_0, A_{m-1} ← A_m, A_7 ← 0） × cnt			×
	SHRW	AX, cnt	2	1	–	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← 0) × cnt			×
	SHL	A, cnt	2	1	–	（CY ← A_7, A_m ← A_{m-1}, A_0 ← 0） × cnt			×
		B, cnt	2	1	–	（CY ← B_7, B_m ← B_{m-1}, B_0 ← 0） × cnt			×
		C, cnt	2	1	–	（CY ← C_7, C_m ← C_{m-1}, C_0 ← 0） × cnt			×
	SHLW	AX, cnt	2	1	–	(CY ← AX_{15}, AX_m ← AX_{m-1}, AX_0 ← 0) × cnt			×
		BC, cnt	2	1	–	(CY ← BC_{15}, BC_m ← BC_{m-1}, BC_0 ← 0) × cnt			×
	SAR	A, cnt	2	1	–	（CY ← A_0, A_{m-1} ← A_m, A_7 ← A_7） × cnt			×
	SARW	AX, cnt	2	1	–	(CY ← AX_0, AX_{m-1} ← AX_m, AX_{15} ← AX_{15}) × cnt			×

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. cnt表示移位个数。
4. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（13/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
循环	ROR	A, 1	2	1	–	（CY, $A_7 \leftarrow A_0$, $A_{m-1} \leftarrow A_m$）× 1			×
	ROL	A, 1	2	1	–	（CY, $A_0 \leftarrow A_7$, $A_{m+1} \leftarrow A_m$）× 1			×
	RORC	A, 1	2	1	–	（CY $\leftarrow A_0$, $A_7 \leftarrow$ CY, $A_{m-1} \leftarrow A_m$）× 1			×
	ROLC	A, 1	2	1	–	（CY $\leftarrow A_7$, $A_0 \leftarrow$ CY, $A_{m+1} \leftarrow A_m$）× 1			×
	ROLWC	AX,1	2	1	–	（CY $\leftarrow AX_{15}$, $AX_0 \leftarrow$ CY, $AX_{m+1} \leftarrow AX_m$）× 1			×
		BC,1	2	1	–	（CY $\leftarrow BC_{15}$, $BC_0 \leftarrow$ CY, $BC_{m+1} \leftarrow BC_m$）× 1			×
位操作	MOV1	CY, saddr.bit	3	1	–	CY ←（saddr）.bit			×
		CY, sfr.bit	3	1	–	CY ← sfr.bit			×
		CY, A.bit	2	1	–	CY ← A.bit			×
		CY, PSW.bit	3	1	–	CY ← PSW.bit			×
		CY,[HL].bit	2	1	4	CY ←（HL）.bit			×
		saddr.bit, CY	3	2	–	（saddr）.bit ← CY			
		sfr.bit, CY	3	2	–	sfr.bit ← CY			
		A.bit, CY	2	1	–	A.bit ← CY			
		PSW.bit, CY	3	4	–	PSW.bit ← CY	×	×	
		[HL].bit, CY	2	2	–	（HL）.bit ← CY			
		CY, ES:[HL].bit	3	2	5	CY ←（ES, HL）.bit			×
		ES:[HL].bit, CY	3	3	–	（ES, HL）.bit ← CY			
	AND1	CY, saddr.bit	3	1	–	CY ← CY ∧（saddr）.bit			×
		CY, sfr.bit	3	1	–	CY ← CY ∧ sfr.bit			×
		CY, A.bit	2	1	–	CY ← CY ∧ A.bit			×
		CY, PSW.bit	3	1	–	CY ← CY ∧ PSW.bit			×
		CY,[HL].bit	2	1	4	CY ← CY ∧（HL）.bit			×
		CY, ES:[HL].bit	3	2	5	CY ← CY ∧（ES, HL）.bit			×
	OR1	CY, saddr.bit	3	1	–	CY ← CY ∨（saddr）.bit			×
		CY, sfr.bit	3	1	–	CY ← CY ∨ sfr.bit			×
		CY, A.bit	2	1	–	CY ← CY ∨ A.bit			×
		CY, PSW.bit	3	1	–	CY ← CY ∨ PSW.bit			×
		CY, [HL].bit	2	1	4	CY ← CY ∨（HL）.bit			×
		CY, ES:[HL].bit	3	2	5	CY ← CY ∨（ES, HL）.bit			×

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（14/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
位操作	XOR1	CY, saddr.bit	3	1	–	CY ← CY ⩛ （saddr）.bit			×
		CY, sfr.bit	3	1	–	CY ← CY ⩛ sfr.bit			×
		CY, A.bit	2	1	–	CY ← CY ⩛ A.bit			×
		CY, PSW.bit	3	1	–	CY ← CY ⩛ PSW.bit			×
		CY, [HL].bit	2	1	4	CY ← CY ⩛ （HL）.bit			×
		CY, ES:[HL].bit	3	2	5	CY ← CY ⩛ （ES, HL）.bit			×
	SET1	saddr.bit	3	2	–	（saddr）.bit ← 1			
		sfr.bit	3	2	–	sfr.bit ← 1			
		A.bit	2	1	–	A.bit ← 1			
		!addr16.bit	4	2	–	（addr16）.bit ← 1			
		PSW.bit	3	4	–	PSW.bit ← 1	×	×	×
		[HL].bit	2	2	–	（HL）.bit ← 1			
		ES:!addr16.bit	5	3	–	（ES, addr16）.bit ← 1			
		ES:[HL].bit	3	3	–	（ES, HL）.bit ← 1			
	CLR1	saddr.bit	3	2	–	（saddr.bit） ← 0			
		sfr.bit	3	2	–	sfr.bit ← 0			
		A.bit	2	1	–	A.bit ← 0			
		!addr16.bit	4	2	–	（addr16）.bit ← 0			
		PSW.bit	3	4	–	PSW.bit ← 0	×	×	×
		[HL].bit	2	2	–	（HL）.bit ← 0			
		ES:!addr16.bit	5	3	–	（ES, addr16）.bit ← 0			
		ES:[HL].bit	3	3	–	（ES, HL）.bit ← 0			
	SET1	CY	2	1	–	CY ← 1			1
	CLR1	CY	2	1	–	CY ← 0			0
	NOT1	CY	2	1	–	CY ← $\overline{CY}$			×

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（15/17）

指令组	助记符	操作数	字节	时钟		操作	标志		
				注 1	注 2		Z	AC	CY
调用/返回	CALL	rp	2	3	–	（SP – 2）← （PC + 2）S, （SP – 3）← （PC + 2）H, （SP – 4）← （PC + 2）L, PC ← CS, rp, SP ← SP – 4			
		$!addr20	3	3	–	（SP – 2）← （PC + 3）S, （SP – 3）← （PC + 3）H, （SP – 4）← （PC + 3）L, PC ← PC + 3 + jdisp16, SP ← SP – 4			
		!addr16	3	3	–	（SP – 2）← （PC + 3）S, (SP – 3）← （PC + 3）H, （SP – 4）← （PC + 3）L, PC ← 0000, addr16, SP ← SP – 4			
		!!addr20	4	3	–	（SP – 2）← （PC + 4）S, （SP – 3）← （PC + 4）H, （SP – 4）← （PC + 4）L, PC ← addr20, SP ← SP – 4			
	CALLT	[addr5]	2	5	–	（SP – 2）← （PC + 2）S, （SP – 3）← （PC + 2）H, （SP – 4）← （PC + 2）L , PCS ← 0000, PCH ← （0000, addr5 + 1）， PCL ← （0000, addr5）， SP ← SP – 4			
	BRK	–	2	5	–	（SP – 1）← PSW, （SP – 2）← （PC + 2）S, （SP – 3）← （PC + 2）H, （SP – 4）← （PC + 2）L, PCS ← 0000, PCH ← （0007FH）, PCL ← （0007EH）， SP ← SP – 4, IE ← 0			
	RET	–	1	6	–	PCL ← （SP）, PCH ← （SP + 1）， PCS ← （SP + 2）, SP ← SP + 4			
	RETI	–	2	6	–	PCL ← （SP）, PCH ← （SP + 1）， PCS ← （SP + 2）, PSW ← （SP + 3）， SP ← SP + 4	R	R	R
	RETB	–	2	6	–	PCL ← （SP）, PCH ← （SP + 1）， PCS ← （SP + 2）, PSW ← （SP + 3）， SP ← SP + 4	R	R	R

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。

备注 **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（16/17）

指令组	助记符	操作数	字节	时钟		操作	标志
				注 1	注 2		Z AC CY
堆栈操作	PUSH	PSW	2	1	–	（SP – 1） ← PSW, （SP – 2） ← 00H, SP ← SP – 2	
		rp	1	1	–	（SP – 1） ← rp_H, （SP – 2）← rp_L, SP ← SP – 2	
	POP	PSW	2	3	–	PSW ← （SP + 1）, SP ← SP + 2	R R R
		rp	1	1	–	rp_L ← （SP）, rp_H ← （SP + 1）, SP ← SP + 2	
	MOVW	SP, #word	4	1	–	SP ← word	
		SP, AX	2	1	–	SP ← AX	
		AX, SP	2	1	–	AX ← SP	
		HL, SP	3	1	–	HL ← SP	
		BC, SP	3	1	–	BC ← SP	
		DE, SP	3	1	–	DE ← SP	
	ADDW	SP, #byte	2	1	–	SP ← SP + byte	
	SUBW	SP, #byte	2	1	–	SP ← SP – byte	
无条件转移	BR	AX	2	3	–	PC ← CS, AX	
		$addr20	2	3	–	PC ← PC + 2 + jdisp8	
		$!addr20	3	3	–	PC ← PC + 3 + jdisp16	
		!addr16	3	3	–	PC ← 0000, addr16	
		!!addr20	4	3	–	PC ← addr20	
条件转移	BC	$addr20	2	2/4[注3]	–	如果CY = 1，PC ← PC + 2 + jdisp8	
	BNC	$addr20	2	2/4[注3]	–	如果CY = 0，PC ← PC + 2 + jdisp8	
	BZ	$addr20	2	2/4[注3]	–	如果Z = 1，PC ← PC + 2 + jdisp8	
	BNZ	$addr20	2	2/4[注3]	–	如果Z = 0，PC ← PC + 2 + jdisp8	
	BH	$addr20	3	2/4[注3]	–	如果（Z ∨ CY） = 0，PC ← PC+3+jdisp8	
	BNH	$addr20	3	2/4[注3]	–	如果（Z ∨ CY） =1，PC ← PC+3+jdisp8	
	BT	saddr.bit, $addr20	4	3/5[注3]	–	如果（saddr）.bit = 1，PC ← PC + 4 + jdisp8	
		sfr.bit, $addr20	4	3/5[注3]	–	如果 sfr.bit = 1，PC ← PC + 4 + jdisp8	
		A.bit, $addr20	3	3/5[注3]	–	如果A.bit = 1，PC ← PC + 3 + jdisp8	
		PSW.bit, $addr20	4	3/5[注3]	–	如果 PSW.bit = 1，PC ← PC + 4 + jdisp8	
		[HL].bit, $addr20	3	3/5[注3]	6/8	如果（HL）.bit = 1，PC ← PC + 3 + jdisp8	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/9	如果（ES, HL）.bit = 1，PC ← PC + 4 + jdisp8	

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. 表示当不符合条件时_/当符合条件时的时钟数。”

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

表 5-5. 操作列表（17/17）

指令组	助记符	操作数	字节	时钟		操作	标志
				注 1	注 2		Z AC CY
条件转移	BF	saddr.bit, $addr20	4	3/5[注3]	–	如果（saddr）.bit = 0，PC ← PC + 4 + jdisp8	
		sfr.bit, $addr20	4	3/5[注3]	–	如果sfr.bit = 0，PC ← PC + 4 + jdisp8	
		A.bit, $addr20	3	3/5[注3]	–	如果A.bit = 0，PC ← PC + 3 + jdisp8	
		PSW.bit, $addr20	4	3/5[注3]	–	如果PSW.bit = 0，PC ← PC + 4 + jdisp8	
		[HL].bit, $addr20	3	3/5[注3]	6/8	如果（HL）.bit = 0，PC ← PC + 3 + jdisp8	
		ES:[HL].bit, $addr20	4	4/6[注3]	7/9	如果（ES, HL）.bit = 0，PC ← PC + 4 + jdisp8	
	BTCLR	saddr.bit, $addr20	4	3/5[注3]	–	如果（saddr）.bit = 1，PC ← PC + 4 + jdisp8 然后复位（saddr）.bit	
		sfr.bit, $addr20	4	3/5[注3]	–	如果sfr.bit = 1，PC ← PC + 4 + jdisp8 然后复位sfr.bit	
		A.bit, $addr20	3	3/5[注3]	–	如果A.bit = 1，PC ← PC + 3 + jdisp8 然后复位A.bit	
		PSW.bit, $addr20	4	5/7[注3]	–	如果PSW.bit = 1，PC ← PC + 4 + jdisp8 然后复位PSW.bit	× × ×
		[HL].bit, $addr20	3	3/5[注3]	–	如果（HL）.bit = 1，PC ← PC + 3 + jdisp8 然后复位（HL）.bit	
		ES:[HL].bit, $addr20	4	4/6[注3]	–	如果（ES, HL）.bit = 1，PC ← PC + 4 + jdisp8 然后复位（ES, HL）.bit	
条件跳转	SKC	–	2	1	–	如果CY = 1跳过下一条指令	
	SKNC	–	2	1	–	如果CY = 0跳过下一条指令	
	SKZ	–	2	1	–	如果Z = 1跳过下一条指令	
	SKNZ	–	2	1	–	如果Z = 0跳过下一条指令	
	SKH	–	2	1	–	如果（Z ∨ CY） = 0跳过下一条指令	
	SKNH	–	2	1	–	如果（Z ∨ CY） = 1跳过下一条指令	
CPU控制	SEL	RBn	2	1	–	RBS[1:0] ← n	
	NOP	–	1	1	–	无操作	
	EI	–	3	4	–	IE ← 1（允许中断）	
	DI	–	3	4	–	IE ← 0（禁止中断）	
	HALT	–	2	3	–	设置HALT模式	
	STOP	–	2	3	–	设置STOP模式	

注 **1.** 当访问内部RAM区域或SFR区域时，或者对于无访问数据的指令。
2. 当访问程序存储器区域时。
3. 表示“当不符合条件时 /当符合条件时”的时钟数。

<R> **备注** **1.** 一个指令时钟周期是指由系统时钟控制寄存器（CKC）选择的CPU 时钟（f_{CLK}） 的一个周期。
2. 该时钟周期用于内部ROM（flash存储器）程序。
3. n表示寄存器bank号（n = 0到3）
4. 在外部存储器区域与内部flash区域相邻的产品中，为了使用外部总线接口功能，等待时钟数与指令执行时钟数相加的值在flash存储器中最后地址（最大16字节）中。因为在预读指令码期间，由于访问超出flash空间的外部存储器区域而插入外部存储器等待时钟。关于等待时钟数，请参考**7.2.2 访问外部存储器数据内容**。

5.6 指令格式

指令包括操作数后固定的操作码。格式如下表所示。

表 5-6. 指令格式列表 （1/30）

助记符	操作数	操作码				
		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	–	–
	A, ES	8E	FD	–	–	–
	ES, A	9E	FD	–	–	–
	CS, #byte	CE	FC	data	–	–

表 5-6. 指令格式列表 （2/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
MOV	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	–
	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	–	–	–

表 5-6. 指令格式列表 （3/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
MOV	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	–	–
	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	–	–

表 5-6. 指令格式列表 （4/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
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	adr1	adrh	–	–
	ES:!addr16	11	F5	adr1	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	–	–	–	–
	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	–	–

表 5-6. 指令格式列表 （5/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
MOVW	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	A4	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	–
	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	–	–	–	–

表 5-6. 指令格式列表 （6/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
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	–
	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	–	–	–

表 5-6. 指令格式列表 （7/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
ADDC	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	–	–	–
	A, L	61	2E	–	–	–
	A, H	61	2F	–	–	–
	X, A	61	20	–	–	–
	A, A	61	21	–	–	–
	C, A	61	30	–	–	–
	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	–	–

表 5-6. 指令格式列表 （8/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
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	–	–	–
	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	–	–	–

表 5-6. 指令格式列表 （9/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
AND	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	–	–
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	–	–

表 5-6. 指令格式列表 （10/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
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	–	–	–
	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	–	–	–

表 5-6. 指令格式列表 （11/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
CMP	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	–	–	–
	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	datal	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	datal	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	–

表 5-6. 指令格式列表 （12/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
CMPW	AX, #word	44	datal	datah	–	–
	AX, BC	43	–	–	–	–
	AX, DE	45	–	–	–	–
	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	–	–	–	–
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	–

表 5-6. 指令格式列表 （13/30）

助记符	操作数	操作码				
		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	–	–	–
	AX, 13	31	DE	–	–	–
	AX, 14	31	EE	–	–	–
	AX, 15	31	FE	–	–	–

表 5-6. 指令格式列表 （14/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
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	_	_	_
	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	_	_	_

表 5-6. 指令格式列表 （15/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
SHLW	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	–	–	–
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	–	–

表 5-6. 指令格式列表 （16/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
MOV1	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	–	–	–
	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	–	–

表 5-6. 指令格式列表 （17/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
MOV1	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	–	–	–
	[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	–	–

表 5-6. 指令格式列表 （18/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
MOV1	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	–	–	–
	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	–	–

表 5-6. 指令格式列表 （19/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
AND1	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	–	–
	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	–	–	–

表 5-6. 指令格式列表 （20/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
OR1	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	–	–
	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	–	–

表 5-6. 指令格式列表 （21/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
XOR1	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	–	–	–
	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	–	–

表 5-6. 指令格式列表 （22/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
SET1	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	–
	!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	–	–	–

表 5-6. 指令格式列表 （23/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
SET1	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	–	–
	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	–	–	–

表 5-6. 指令格式列表 （24/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
CLR1	!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	–	–	–
	[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	–	–	–

表 5-6. 指令格式列表 （25/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
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	–	–	–
	[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	–	–	–

表 5-6. 指令格式列表 （26/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
PUSH	PSW	61	DD	–	–	–
	AX	C1	–	–	–	–
	BC	C3	–	–	–	–
	DE	C5	–	–	–	–
	HL	C7	–	–	–	–
POP	PSW	61	CD	–	–	–
	AX	C0	–	–	–	–
	BC	C2	–	–	–	–
	DE	C4	–	–	–	–
	HL	C6	–	–	–	–
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	–

表 5-6. 指令格式列表 （27/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
BT	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	–	–
	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	–

表 5-6. 指令格式列表 （28/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
BF	sfr.0, $addr20	31	84	sfr	adr	–
	sfr.1, $addr20	31	94	sfr	adr	–
	sfr.2, $addr20	31	A4	sfr	adr	–
	sfr.3,$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	–

表 5-6. 指令格式列表 （29/30）

助记符	操作数	操作码				
		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	–	–
	[HL].7, $addr20	31	F1	adr	–	–

表 5-6. 指令格式列表 （30/30）

助记符	操作数	操作码				
		1st	2nd	3rd	4th	5th
BTCLR	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	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	_	_	_	_

5.7 指令映射图

指令映射图如表 5-7到 5-10 所示。

表 5-7. 指令映射图（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,!addr16	ADDW AX,BC	ADDW AX,#word	ADDW AX,DE	ADDW AX,saddrp	ADDW AX,HL	XCH A,X	MOV A,word[B]	ADD saddr,#byte	ADD A,saddr	ADD A,#byte	ADD A,[HL]	ADD A,[HL+byte]	ADD A,!addr16
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[B],A	MOV word[B],#byte	ADDC saddr,#byte	ADDC A,saddr	ADDC A,#byte	ADDC A,[HL]	ADDC A,[HL+byte]	ADDC A,!addr16
2	SUBW SP,#byte		SUBW AX,!addr16	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,!addr16
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,!addr16
4	CMP !addr16,#byte	MOV ES,#byte	CMPW AX,!addr16	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,!addr16
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,!addr16
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,!addr16
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,!addr16
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,!addr16
9	DEC X	DEC A	DEC C	DEC B	DEC E	DEC D	DEC L	DEC H	MOV [SP+byte],A	MOV [DE],A	MOV [DE+byte],A	MOV [HL],A	MOV [HL+byte],A	MOV saddr,A	MOV sfr,A	MOV !addr16,A
a	INC !addr16	INCW AX	INCW !addr16	INCW BC	INC saddr	INCW DE	INCW saddrp	INCW HL	MOVW AX,[SP+byte]	MOVW AX,[DE]	MOVW AX,[DE+byte]	MOVW AX,[HL]	MOVW AX,[HL+byte]	MOVW AX,saddrp	MOVW AX,sfrp	MOVW AX,!addr16
b	DEC !addr16	DECW AX	DECW !addr16	DECW BC	DEC saddr	DECW DE	DECW saddrp	DECW HL	MOVW [SP+byte],AX	MOVW [DE],AX	MOVW [DE+byte],AX	MOVW [HL],AX	MOVW [HL+byte],AX	MOVW saddrp,AX	MOVW sfrp,AX	MOVW !addr16,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	MOV sfr,#byte	MOV !addr16,#byte
d	CMP0 X	CMP0 A	CMP0 C	CMP0 B	CMP0 saddr	CMP0 !addr16	MULU X	RET	MOV X,saddr	MOV X,!addr16	MOVW BC,saddrp	MOVW BC,!addr16	BC $addr20	BZ $addr20	BNC $addr20	BNZ $addr20
e	ONEB X	ONEB A	ONEB C	ONEB B	ONEB saddr	ONEB !addr16	ONEW AX	ONEW BC	MOV B,saddr	MOV B,!addr16	MOVW DE,saddrp	MOVW DE,!addr16	BR !!addr20	BR !addr16	BR $!addr20	BR $addr20
f	CLRB X	CLRB A	CLRB C	CLRB B	CLRB saddr	CLRB !addr16	CLRW AX	CLRW BC	MOV C,saddr	MOV C,!addr16	MOVW HL,saddrp	MOVW HL,!addr16	CALL !!addr20	CALL !addr16	CALL $!addr20	

表 5-8. 指令映射图 （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,!addr16	XCH A,sfr	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	MOVS [HL+byte],X	SEL RB0
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	ROLC A,1	PUSH PSW	CMPS X,[HL+byte]	SEL RB1
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 RB2
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 RB3

表 5-9. 指令映射图 （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 !addr16.0**	**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 !addr16.0**	**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 !addr16.1**	**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 !addr16.1**	**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 !addr16.2**	**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 !addr16.2**	**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 !addr16.3**	**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 !addr16.3**	**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 !addr16.4**	**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 !addr16.4**	**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 !addr16.5**	**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 !addr16.5**	**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 !addr16.6**	**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 !addr16.6**	**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 !addr16.7**	**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 !addr16.7**	**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**

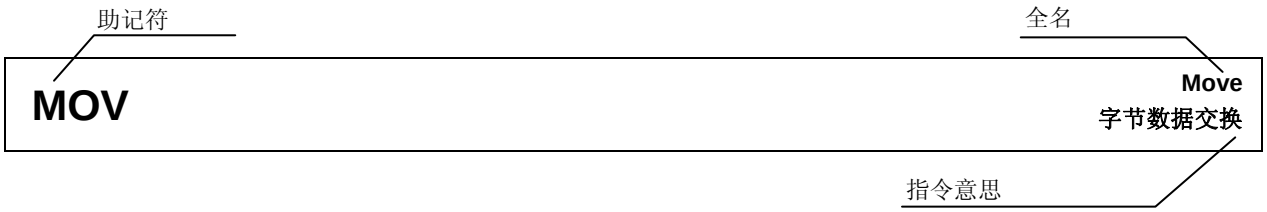
表 5-10. 指令映射图 （4th MAP）

	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,$addr20	BTCLR [HL].0,$addr20	BT sfr.0,$addr20	BT [HL].0,$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,$addr20	BTCLR [HL].1,$addr20	BT sfr.1,$addr20	BT [HL].1,$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,$addr20	BTCLR [HL].2,$addr20	BT sfr.2,$addr20	BT [HL].2,$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,$addr20	BTCLR [HL].3,$addr20	BT sfr.3,$addr20	BT [HL].3,$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,$addr20	BTCLR [HL].4,$addr20	BT sfr.4,$addr20	BT [HL].4,$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,$addr20	BTCLR [HL].5,$addr20	BT sfr.5,$addr20	BT [HL].5,$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,$addr20	BTCLR [HL].6,$addr20	BT sfr.6,$addr20	BT [HL].6,$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,$addr20	BTCLR [HL].7,$addr20	BT sfr.7,$addr20	BT [HL].7,$addr20	BF sfr.7,$addr20	BF [HL].7,$addr20							SHLW BC,15	SHLW AX,15	SHRW AX,15	SARW AX,15

第六章　指令描述

本章解释了78K0R微控制器的指令。

描述示例



[指令格式]　　MOV dst, src：表示指令的基本描述格式。

[操作]　　**dst ← src：**用符号表示指令的操作。

[操作数]　　表示可被该指令指定的操作数。请参考 **5.2 操作栏描述** 中每个操作数符号的描述。

助记符	操作数 （dst, src）
MOV	r, #byte
	A, saddr
	saddr, A
	PSW, #byte

助记符	操作数 （dst, src）
MOV	A, PSW
	[HL], A
	A, [HL+byte]
	[HL+C], A

[标志]　　表示指令执行改变的标志操作。
每个标志操作符在下表中列出。

Z	AC	CY

符号表

符号	描述
空白	不改变
0	清零
1	设置为1
×	根据结果设置或清零
R	存储以前保存的值

[描述]：详细描述指令操作。

- 第二个操作数指定的源操作数（src）的内容传输到第一个操作数指定的目的操作数（dst） 中。

[描述示例]

MOV A, #4DH; 4DH传输到 A 寄存器。

6.1 8位数据传输指令

如下指令是8位数据传输指令。

MOV

Move
字节数据传输

[指令格式] MOV dst, src

[操作] dst ← src

[操作数]

助记符	操作数 （dst, src）
MOV	r, #byte
	saddr, #byte
	sfr, #byte
	!addr16, #byte
	A, r 注
	r, A 注
	A, saddr
	saddr, A
	A, sfr
	sfr, A
	A, !addr16
	!addr16, A
	PSW, #byte
	A, PSW
	PSW, A
	ES, #byte
	ES, saddr
	A, ES
	ES, A
	CS, #byte
	A, CS
	CS, A
	A, [DE]
	[DE], A
	[DE+byte], #byte
	A, [DE+byte]
	[DE+byte], A
	A, [HL]

助记符	操作数 （dst, src）
MOV	[HL], A
	[HL+byte], #byte
	A, [HL+byte]
	[HL+byte], A
	A, [HL+B]
	[HL+B], A
	A, [HL+C]
	[HL+C], A
	word[B], #byte
	A, word[B]
	word[B], A
	word[C], #byte
	A, word[C]
	word[C], A
	word[BC], #byte
	A, word[BC]
	word[BC], A
	[SP+byte], #byte
	A, [SP+byte]
	[SP+byte], A
	B, saddr
	B, !addr16
	C, saddr
	C, !addr16
	X, saddr
	X, !addr16
	ES:!addr16, #byte
	A, ES:!addr16

助记符	操作数 （dst, src）
MOV	ES:!addr16, A
	A, ES:[DE]
	ES:[DE], A
	ES:[DE+byte], #byte
	A, ES:[DE+byte]
	ES:[DE+byte], A
	A, ES:[HL]
	ES:[HL], A
	ES:[HL+byte], #byte
	A, ES:[HL+byte]
	ES:[HL+byte], A
	A, ES:[HL+B]
	ES:[HL+B], A
	A, ES:[HL+C]
	ES:[HL+C], A
	ES:word[B], #byte
	A, ES:word[B]
	ES:word[B], A
	ES:word[C], #byte
	A, ES:word[C]
	ES:word[C], A
	ES:word[BC], #byte
	A, ES:word[BC]
	ES:word[BC], A
	B, ES:!addr16
	C, ES:!addr16
	X, ES:!addr16

注 r = A 除外

[标志]

PSW, #byte 和 PSW, A 操作数

Z	AC	CY
×	×	×

所有其它操作数组合

Z	AC	CY

[描述]

- 第二个操作数指定的源操作数（src）的内容传输到第一个操作数指定的目的操作数（dst） 中。
- 在MOV PSW, #byte 指令/MOV PSW, A 指令和下一个指令之间无中断响应。

[描述示例]

MOV A, #4DH; 4DH传输到 A 寄存器。

XCH

Exchange
字节数据交换

[指令格式] XCH dst, src

[操作] dst ↔ src

[操作数]

助记符	操作数 （dst, src）
XCH	A, r 注
	A, saddr
	A, sfr
	A, !addr16
	A, [DE]
	A, [DE+byte]
	A, [HL]
	A, [HL+byte]
	A, [HL+B]

助记符	操作数 （dst, src）
XCH	A, [HL+C]
	A, ES:!addr16
	A, ES:[DE]
	A, ES:[DE+byte]
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY

[描述]

- 第一个和第二个操作数内容交换。

[描述示例]

XCH A, FFEBCH; A 寄存器内容和地址FFEBCH 中的内容交换。

ONEB

One byte
设置字节数据为01H

[指令格式] ONEB dst

[操作] dst ← 01H

[操作数]

助记符	操作数 （dst）
ONEB	A
	X
	B
	C
	saddr
	!addr16
	ES:!addr16

[标志]

Z	AC	CY

[描述]

- 01H 传输到第一操作数指定的目的操作数（dst）中。

[描述示例]

ONEB A; 传输01H 到 A 寄存器。

CLRB

Clear byte
字节数据清零

[指令格式] CLRB dst

[操作] dst ← 00H

[操作数]

助记符	操作数 （dst）
CLRB	A
	X
	B
	C
	saddr
	!addr16
	ES:!addr16

[标志]

Z	AC	CY

[描述]

- 00H 传输到第一操作数指定的目的操作数（dst） 中。

[描述示例]

CLRB A; 传输00H 到 A 寄存器。

MOVS

Move and change PSW
传输字节数据并改变PSW

[指令格式] MOVS dst, src

[操作] dst ← src

[操作数]

助记符	操作数 （dst, src）
MOVS	[HL+byte], X
	ES:[HL+byte], X

[标志]

Z	AC	CY
×		×

[描述]

- 第二个操作数指定的源操作数（src）的内容传输到第一个操作数指定的目的操作数（dst） 中。
- 如果源操作数的值是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果寄存器 A 的值为0 或者如果源操作数的值为0，CY 标志置位（1）。其它情况下，CY 标志清零（0）。

[描述示例]

MOVS [HL+2H], X; 当 HL = FE00H，X = 55H，A = 0H

“X = 55H”存储在地址FE02H。

Z 标志 = 0

CY 标志 = 1 （A 寄存器 = 0）

6.2 16位数据传输指令

如下指令是16位数据传输指令。

MOVW

Move Word
字数据传输

[指令格式] MOVW dst, src

[操作] dst ← src

[操作数]

助记符	操作数 （dst, src）
MOVW	rp, #word
	saddrp, #word
	sfrp, #word
	AX, saddrp
	saddrp, AX
	AX, sfrp
	sfrp, AX
	AX, rp 注
	rp, AX 注
	AX, !addr16
	!addr16, AX
	AX, [DE]
	[DE], AX
	AX, [DE+byte]
	[DE+byte], AX
	AX, [HL]
	[HL], AX
	AX, [HL+byte]
	[HL+byte], AX
	AX, word[B]
	word[B], AX
	AX, word[C]
	word[C], AX
	AX, word[BC]
	word[BC], AX
	AX, [SP+byte]

助记符	操作数 （dst, src）
MOVW	[SP+byte], AX
	BC, saddrp
	BC, !addr16
	DE, saddrp
	DE, !addr16
	HL, saddrp
	HL, !addr16
	AX, ES:!addr16
	ES:!addr16, AX
	AX, ES:[DE]
	ES:[DE], AX
	AX, ES:[DE+byte]
	ES:[DE+byte], AX
	AX, ES:[HL]
	ES:[HL], AX
	AX, ES:[HL+byte]
	ES:[HL+byte], AX
	AX, ES:word[B]
	ES:word[B], AX
	AX, ES:word[C]
	ES:word[C], AX
	AX, ES:word[BC]
	ES:word[BC], AX
	BC, ES:!addr16
	DE, ES:!addr16
	HL, ES:!addr16

注 仅当rp = BC、DE 或 HL

[标志]

Z	AC	CY

[描述]

- 第二个操作数指定的源操作数（src）的内容传输到第一个操作数指定的目的操作数（dst） 中。

[描述示例]

MOVW AX, HL; HL寄存器的内容传输到AX寄存器中。

[注意事项]

仅可指定偶地址。不能指定奇地址。

XCHW

Exchange Word
字数据交换

[指令格式] XCHW dst, src

[操作] dst ↔ src

[操作数]

助记符	操作数 （dst, src）
XCHW	AX, rp 注

注 仅当rp = BC、DE 或 HL

[标志]

Z	AC	CY

[描述]

- 第一个和第二个操作数内容交换。

[描述示例]

XCHW AX, BC; AX 寄存器和BC寄存器的存储器内容交换。

ONEW

One Word
设置字数据为0001

[指令格式] ONEW dst

[操作] dst ←0001H

[操作数]

助记符	操作数 （dst）
ONEW	AX
	BC

[标志]

Z	AC	CY

[描述]

- 0001H 传输到第一操作数指定的目的操作数（dst）。

[描述示例]

ONEW AX; 0001H 传输到AX寄存器。

CLRW

Clear Word
字数据清零

[指令格式] CLRW dst

[操作] dst ←0000H

[操作数]

助记符	操作数 （dst）
CLRW	AX
	BC

[标志]

Z	AC	CY

[描述]

- 0000H 传输到第一操作数指定的目的操作数（dst）。

[描述示例]

CLRW AX; 0000H 传输到AX寄存器。

6.3 8位操作指令

如下是8位操作指令。

ADD

ADD
字节数据加

[指令格式] ADD dst, src

[操作] dst, CY ← dst + src

[操作数]

助记符	操作数 （dst, src）
ADD	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
ADD	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）和第二个操作数指定的源操作数（src）的内容相加，结果存储在CY标志和目的操作数（dst） 中。
- 如果加的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果加法从第7位产生一个进位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 如果加法从第3位到第4位产生一个进位，AC标志被设置为 （1）。其它情况下，AC标志清零（0）。

[描述示例]

ADD CR10, #56H; 56H和CR10寄存器的值相加，结果存储在CR10 寄存器中。

ADDC

Add with Carry
带进位的字节数据加法

[指令格式] ADDC dst, src

[操作] dst, CY ← dst + src + CY

[操作数]

助记符	操作数 （dst, src）
ADDC	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
ADDC	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）和第二个操作数指定的源操作数（src）的内容并且加上CY，结果存储在CY标志和目的操作数（dst） 中。
  CY标志加到LSB。此指令主要用于两个或更多字节的加法。
- 如果加的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果加法从第7位产生一个进位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 如果加法从第3位到第4位产生一个进位，AC标志被设置为 （1）。其它情况下，AC标志清零（0）。

[描述示例]

ADDC A, [HL+B]; A寄存器内容和（HL寄存器+（B寄存器））地址处内容及CY标志相加，结果保存在A寄存器中。

SUB

Subtract
字节数据减

[指令格式] SUB dst, src

[操作] dst, CY ← dst – src

[操作数]

助记符	操作数 （dst, src）
SUB	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
SUB	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去第二个操作数指定的源操作数（src），结果存储在目的操作数（dst） 和CY标志。
  如果源操作数（src）和目的操作数（dst）相等，目的操作数可以清零。
- 如果减的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果减法在第7位产生一个借位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 如果减法在从第4位到第3位产生一个借位，AC标志被置（1）。其它情况下，AC标志清零（0）。

[描述示例]

SUB D, A; A 寄存器从D寄存器中减去，并且结果存储在D寄存器中。

SUBC

Subtract with Carry
带借位的字节数据减

[指令格式] SUBC dst, src

[操作] dst, CY ← dst – src – CY

[操作数]

助记符	操作数 （dst, src）
SUBC	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
SUBC	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去CY标志和第二个操作数指定的源操作数（src）的内容，结果存储在目的操作数（dst） 中。
  从LSB位减CY标志。此指令主要用于两个或更多字节的减法。
- 如果减的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果减法在第7位产生一个借位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 如果减法在从第4位到第3位产生一个借位，AC标志被置（1）。其它情况下，AC标志清零（0）。

[描述示例]

SUBC A, [HL]; A寄存器内容减去（HL寄存器）地址处内容和CY标志，结果存在A寄存器中。

AND

And
字节数据的逻辑与

[指令格式] AND dst, src

[操作] dst ← dst ∧ src

[操作数]

助记符	操作数 （dst, src）
AND	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
AND	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×		

[描述]

- 第一个操作数指定的目的操作数（dst）和第二个操作数指定的源操作数（src）的内容以位相与，结果存储在目的操作数（dst） 中。
- 如果逻辑乘积显示所有的位是0，Z 标志被设置为（1）。其它情况下，Z 标志清零 （0）。

[描述示例]

AND FFEBAH, #11011100B; FFEBAH内容与11011100B位与结果存于FFEBAH中。

OR

Or
字节数据的逻辑或

[指令格式] OR dst, src

[操作] dst ← dst ∨ src

[操作数]

助记符	操作数 （dst, src）
OR	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
OR	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×		

[描述]

- 第一个操作数指定的目的操作数（dst）和第二个操作数指定的源操作数（src）的内容以位相或，结果存储在目的操作数（dst） 中。
- 如果逻辑和显示所有的位是0，Z 标志被设置为（1）。其它情况下，Z 标志清零 （0）。

[描述示例]

OR A, FFE98H; A 寄存器和 FFE98H以位相或，结果存储在A寄存器中。

XOR

Exclusive Or
字节数据的逻辑异或

[指令格式] XOR dst, src

[操作] dst ← dst ⊻ src

[操作数]

助记符	操作数 （dst, src）
XOR	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]

助记符	操作数 （dst, src）
XOR	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]

注 r = A 除外

[标志]

Z	AC	CY
×		

[描述]

- 第一个操作数指定的目的操作数（dst）和第二个操作数指定的源操作数（src）的内容以位相异或，结果存储在目的操作数（dst） 中。
  用此指令选择#0FFH 为源操作数（src）可使目的操作数（dst）的所有位逻辑反。
- 如果逻辑异和显示所有的位是0，Z 标志被设置为（1）。其它情况下，Z 标志清零 （0）。

[描述示例]

XOR A, L; A 和L 寄存器以位相异或，结果存储在A寄存器中。

CMP

Compare
字节数据比较

[指令格式] CMP dst, src

[操作] dst – src

[操作数]

助记符	操作数 （dst, src）
CMP	A, #byte
	saddr, #byte
	A, r 注
	r, A
	A, saddr
	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]

助记符	操作数 （dst, src）
CMP	A, [HL+C]
	!addr16, #byte
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]
	ES:!addr16, #byte

注 r = A 除外

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去第二个操作数指定的源操作数（src）。
  减的结果不存储，只有Z、AC和CY标志改变。
- 如果减的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果减法在第7位产生一个借位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 如果减法在从第4位到第3位产生一个借位，AC标志被置（1）。其它情况下，AC标志清零（0）。

[描述示例]

CMP FFE38H, #38H; 地址FFE38H处内容与38H相减，仅标志改变。
（地址FFE38H处内容与立即数比较）。

CMP0

Compare 00H
字节数据与0比较

[指令格式] CMP0 dst

[操作] dst – 00H

[操作数]

助记符	操作数 （dst）
CMP0	A
	X
	B
	C
	saddr
	!addr16
	ES:!addr16

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去00H。
- 减的结果不存储，只有Z、AC和CY标志改变。
- 如果目的操作数是 0，Z 标志被置为（1）。其它情况下，Z 标志清零 （0）。
- AC和CY 标志一直为0。

[描述示例]

CMP0 A; 如果A 寄存器的值为0，Z 标志置位。

CMPS

Compare
字节数据比较

[指令格式] CMPS dst, src

[操作] dst – src

[操作数]

助记符	操作数 （dst, src）
CMPS	X, [HL+byte]
	X, ES:[HL+byte]

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去第二个操作数指定的源操作数（src）。
  减的结果不存储，只有Z、AC和CY标志改变。
- 如果减的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 当计算结果不为0或当寄存器A或目的操作数的值为0时，CY 标志置 （1）。其它情况下，CY 标志清零（0）。
- 如果减法在从第4位到第3位产生一个借位，AC标志被置（1）。其它情况下，AC标志清零（0）。

[描述示例]

CMPS X, [HL+F0H]; 当 HL = FD12H

X的值与地址FFE02H的内容比较，如果两个值相等Z 标志置1。
X的值与地址FFE02H的内容比较，如果两个值不相等CY 标志置1。
当寄存器A 的值为0时，CY 标志置1。
当寄存器X 的值为0时，CY 标志置1。
从第4位到第3位产生一个借位，AC标志被置（1）， 与CMP指令相似。

6.4 16位操作指令

如下是16位操作指令。

ADDW

Add Word
字数据加

[指令格式] ADDW dst, src

[操作] dst, CY ← dst + src

[操作数]

助记符	操作数 （dst, src）
ADDW	AX, #word
	AX, AX
	AX, BC
	AX, DE
	AX, HL
	AX, saddrp
	AX, !addr16
	AX, [HL+byte]
	AX, ES:!addr16
	AX, ES:[HL+byte]

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）和第二个操作数指定的源操作数（src）的内容相加，结果存储在目的操作数（dst） 中。
- 如果加的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果加法从第15位产生一个进位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 作为加法的结果，AC标志不确定。

[描述示例]

ADDW AX, #ABCDH; ABCDH和AX寄存器的值相加，结果存储在AX寄存器中。

SUBW

Subtract Word
字数据减

[指令格式] SUBW dst, src

[操作] dst, CY ← dst – src

[操作数]

助记符	操作数 （dst, src）
SUBW	AX, #word
	AX, BC
	AX, DE
	AX, HL
	AX, saddrp
	AX, !addr16
	AX, [HL+byte]
	AX, ES:!addr16
	AX, ES:[HL+byte]

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去第二个操作数指定的源操作数（src），结果存储在目的操作数（dst） 和CY标志。
- 如果减的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果减法在第15位产生一个借位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 作为减法的结果，AC标志不确定。

[描述示例]

SUBW AX, #ABCDH; AX寄存器的值减去ABCDH，结果存储在AX寄存器中。

CMPW

Compare Word
字数据比较

[指令格式] CMPW dst, src

[操作] dst – src

[操作数]

助记符	操作数 （dst, src）
CMPW	AX, #word
	AX, BC
	AX, DE
	AX, HL
	AX, saddrp
	AX, !addr16
	AX, [HL+byte]
	AX, ES:!addr16
	AX, ES:[HL+byte]

[标志]

Z	AC	CY
×	×	×

[描述]

- 第一个操作数指定的目的操作数（dst）减去第二个操作数指定的源操作数（src）。
  减的结果不存储，只有Z、AC和CY标志改变。
- 如果减的结果显示dst 是0，Z 标志设置为（1）。其它情况下，Z 标志清零 （0）。
- 如果减法在第15位产生一个借位，CY标志被设置为 （1）。其它情况下，CY 标志清零（0）。
- 作为减法的结果，AC标志不确定。

[描述示例]

CMPW AX, #ABCDH; AX寄存器减去ABCDH，仅标志改变（AX寄存器与立即数比较） 。

6.5 乘法指令

如下是乘法指令。

MULU ... 120

MULU

Multiply Unsigned
无符号数据乘法

[指令格式] MULU src

[操作] AX ← A × src

[操作数]

助记符	操作数 （src）
MULU	X

[标志]

Z	AC	CY

[描述]

- A寄存器的值与源操作数（src）以无符号数据相乘，结果存储在AX寄存器中。

[描述示例]

MULU X; A寄存器的值与X 寄存器的值相乘，结果存储在AX寄存器中。

6.6 加 / 减指令

如下是加/减指令。

INC

加
字节数据加

[指令格式] INC dst

[操作] dst ← dst + 1

[操作数]

助记符	操作数 （dst）
INC	r
	saddr
	!addr16
	[HL+byte]
	ES:!addr16
	ES:[HL+byte]

[标志]

Z	AC	CY
×	×	

[描述]

- 目的操作数 （dst） 的内容增加一。
- 如果增加结果是0，Z 标志置为（1）。其它情况下，Z 标志清零 （0）。
- 如果增加从第3位到第4位产生一个进位，AC标志置为（1）。其它情况下，AC标志清零（0）。
- 因为该指令对一个重复的操作会频繁使用，CY标志的内容不改变（多字节操作时保持CY标志内容）。

[描述示例]

INC B; B 寄存器增加。

DEC

Decrement
字节数据减

[指令格式] DEC dst

[操作] dst ← dst – 1

[操作数]

助记符	操作数 （dst）
DEC	r
	saddr
	!addr16
	[HL+byte]
	ES:!addr16
	ES:[HL+byte]

[标志]

Z	AC	CY
×	×	

[描述]

- 目的操作数 （dst） 的内容减一。
- 如果减的结果是0，Z 标志置为（1）。其它情况下，Z 标志清零 （0）。
- 如果递减从第4位到第3位产生一个借位，AC标志置为（1）。其它情况下，AC标志清零（0）。
- 因为该指令对一个重复的操作会频繁使用，CY标志的内容不改变（多字节操作时保持CY标志内容）。
- 如果dst 是 B 或C 寄存器或者saddr，并且不希望改变AC和CY标志的内容，可以使用DBNZ 指令。

[描述示例]

DEC FFE92H; 地址FFE92H的内容递减。

INCW

Increment Word
字数据加

[指令格式] INCW dst

[操作] dst ← dst + 1

[操作数]

助记符	操作数 （dst）
INCW	rp
	saddrp
	!addr16
	[HL+byte]
	ES:!addr16
	ES:[HL+byte]

[标志]

Z	AC	CY

[描述]

- 目的操作数 （dst） 的内容增加一。
- 因为该指令对一个用作寻址的寄存器（指针）的递增会频繁使用，Z 、AC和CY标志的内容不改变。

[描述示例]

INCW HL; HL 寄存器递增。

DECW

Decrement Word
字数据减

[指令格式] DECW dst

[操作] dst ← dst – 1

[操作数]

助记符	操作数 （dst）
DECW	rp
	saddrp
	!addr16
	[HL+byte]
	ES:!addr16
	ES:[HL+byte]

[标志]

Z	AC	CY

[描述]

- 目的操作数 （dst） 的内容减一。
- 因为该指令对一个用作寻址的寄存器（指针）的递减会频繁使用，Z、AC和CY标志的内容不改变。

[描述示例]

DECW DE; DE寄存器递减。

6.7 移位指令

如下是移位指令。

SHR

Shift Right
逻辑右移

[指令格式] SHR dst, cnt

[操作] （CY ← dst_0, dst_{m-1} ← dst_m, dst_7 ← 0） × cnt

[操作数]

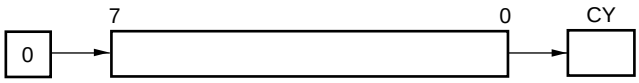
助记符	操作数 （dst, cnt）
SHR	A, cnt

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容右移由cnt指定的次数。
- “0”移至MSB（第7位）并且第0位的值移至CY。
- cnt 可以指定为从1到7的任何值。



[描述示例]

SHR A, 3; 当A 寄存器的值为F5H时，A = 1EH 和 CY = 1。

A = 1111_0101B　CY = 0
A = 0111_1010B　CY = 1　第1次
A = 0011_1101B　CY = 0　第2次
A = 0001_1110B　CY = 1　第3次

SHRW

Shift Right Word
逻辑右移

[指令格式] SHRW dst, cnt

[操作] （CY ← dst_0, dst_{m-1} ← dst_m, dst_{15} ← 0） × cnt

[操作数]

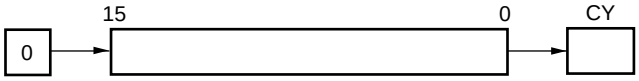
助记符	操作数 （dst, cnt）
SHRW	AX, cnt

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容右移由cnt指定的次数。
- “0”移至MSB（第15位）并且第0位的值移至CY。
- cnt 可以指定为从1到15的任何值。



[描述示例]

SHRW AX 3; 当AX 寄存器的值为AAF5H时，AX = 155EH 和 CY = 1。

AX = 1010_1010_1111_0101B CY = 0
AX = 0101_0101_0111_1010B CY = 1 第1次
AX = 0010_1010_1011_1101B CY = 0 第2次
AX = 0001_0101_0101_1110B CY = 1 第3次

SHL

Shift Left
逻辑左移

[指令格式] SHL dst, cnt

[操作] （CY ← dst_7, dst_m ← dst_{m-1}, dst_0 ← 0） × cnt

[操作数]

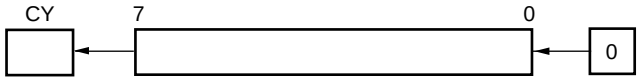
助记符	操作数 （dst, cnt）
SHL	A, cnt
	B, cnt
	C, cnt

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容左移由cnt指定的次数。
- “0”移至LSB（第0位）并且第7位的值移至CY。
- cnt 可以指定为从1到7的任何值。



[描述示例]

SHL A, 3; 当A 寄存器的值为5DH时，A = E8H 和 CY = 0。

CY = 0　A = 0101_1101B
CY = 0　A = 1011_1010B　第1次
CY = 1　A = 0111_0100B　第2次
CY = 0　A = 0110_1000B　第3次

SHLW

Shift Left Word
逻辑左移

[指令格式] SHLW dst, cnt

[操作] （CY ← dst_{15}, dst_m ← dst_{m-1}, dst_0 ← 0） × cnt

[操作数]

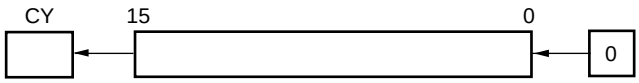
助记符	操作数 （dst, cnt）
SHLW	AX, cnt
	BC, cnt

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容左移由cnt指定的次数。
- “0”移至LSB（第0位）并且第15位的值移至CY。
- cnt 可以指定为从1到15的任何值。



[描述示例]

SHLW BC, 3; 当BC 寄存器的值为C35DH时，BC = 1AE8H 和 CY = 0。

CY = 0 BC = 1100_0011_0101_1101B
CY = 1 BC = 1000_0110_1011_1010B 第1次
CY = 1 BC = 0000_1101_0111_0100B 第2次
CY = 0 BC = 0001_1010_1110_1000B 第3次

SAR

Shift Arithmetic Right
算术右移

[指令格式] SAR dst, cnt

[操作] （CY ← dst_0, dst_{m-1} ← dst_m, dst_7 ← dst_7） × cnt

[操作数]

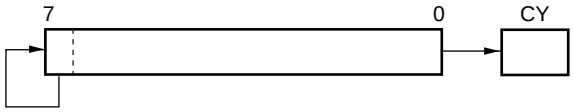
助记符	操作数 （dst, cnt）
SHR	A, cnt

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容右移由cnt指定的次数。
- 相同的值保存在MSB（第7位）中并且第0位的值移至CY。
- cnt 可以指定为从1到7的任何值。



[描述示例]

SAR A, 4; 当A 寄存器的值为8CH时，A = F8H 和 CY = 1。

A = 1000_1100B CY = 0
A = 1100_0110B CY = 0 第1次
A = 1110_0011B CY = 0 第2次
A = 1111_0001B CY = 1 第3次
A = 1111_1000B CY = 1 第4次

SARW

Shift Arithmetic Right Word
算术右移

[指令格式] SARW dst, cnt

[操作] （CY ← dst_0, dst_{m-1} ← dst_m, dst_{15} ← dst_{15}） × cnt

[操作数]

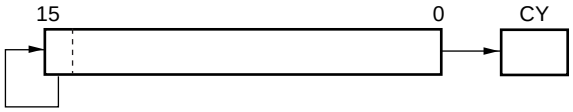
助记符	操作数 （dst, cnt）
SARW	AX, cnt

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容右移由cnt指定的次数。
- 相同的值保存在MSB（第15位）中并且第0位的值移至CY。
- cnt 可以指定为从1到15的任何值。



[描述示例]

SAR AX, 4; 当AX 寄存器的值为A28CH时，AX = FA28H 和 CY = 1。

AX = 1010_0010_1000_1100B CY = 0
AX = 1101_0001_0100_0110B CY = 0 第1次
AX = 1110_1000_1010_0011B CY = 0 第2次
AX = 1111_0100_0101_0001B CY = 1 第3次
AX = 1111_1010_0010_1000B CY = 1 第4次

6.8 循环指令

如下是循环指令。

ROR

Rotate Right
字节数据右移

[指令格式] ROR dst, cnt

[操作] （CY, $dst_7 \leftarrow dst_0$, $dst_{m-1} \leftarrow dst_m$） × 1次

[操作数]

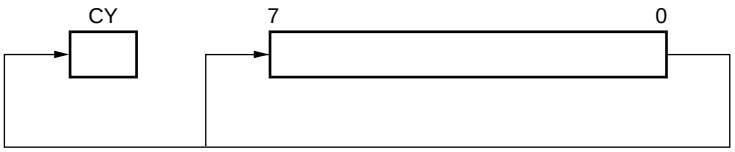
助记符	操作数 （dst, cnt）
ROR	A, 1

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容右移一次。
- LSB （第0位） 的内容同时移至MSB （第7位）并且传输给CY 标志。



[描述示例]

ROR A, 1; A 寄存器内容右移一位。

ROL

Rotate Left
字节数据左移

[指令格式] ROL dst, cnt

[操作] （CY, $dst_0 \leftarrow dst_0$, $dst_{m+1} \leftarrow dst_m$） × 1次

[操作数]

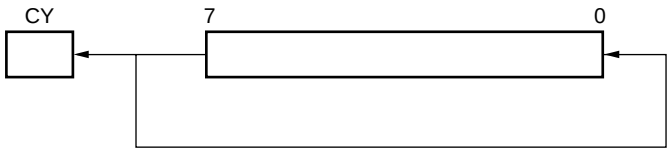
助记符	操作数 （dst, cnt）
ROL	A, 1

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数（dst） 的内容左移一次。
- MSB （第7位） 的内容同时移至LSB （第0位）并且传输给CY 标志。



[描述示例]

ROL A, 1; A 寄存器内容左移一位。

RORC

Rotate Right with Carry
字节数据带进位右移

[指令格式] RORC dst, cnt

[操作] （CY ← dst_0, dst_7 ← CY, dst_{m-1} ← dst_m） × 1次

[操作数]

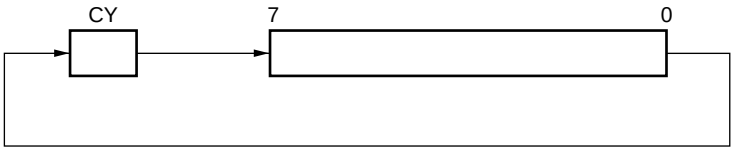
助记符	操作数 （dst, cnt）
RORC	A, 1

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数 （dst） 内容仅右移一次包括CY 标志。



[描述示例]

RORC A, 1; A寄存器内容包括CY 标志右移一位。

ROLC

Rotate Left with Carry
字节数据带进位左移

[指令格式] ROLC dst, cnt

[操作] （CY ← dst_7, dst_0 ← CY, dst_{m+1} ← dst_m） × 1次

[操作数]

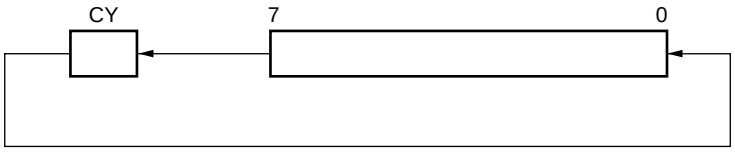
助记符	操作数 （dst, cnt）
ROLC	A, 1

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数 （dst） 内容仅左移一次包括CY 标志。



[描述示例]

ROLC A, 1; A寄存器内容包括CY 标志左移一位。

ROLWC

Rotate Left word with Carry
带进位字数据左移

[指令格式] ROLWC dst, cnt

[操作] （CY ← dst_{15}, dst_0 ← CY, dst_{m+1} ← dst_m） × 1次

[操作数]

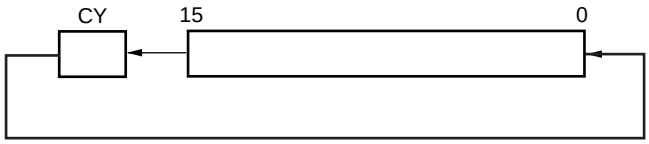
助记符	操作数 （dst, cnt）
ROLWC	AX, 1
	BC, 1

[标志]

Z	AC	CY
		×

[描述]

- 第一个操作数指定的目的操作数 （dst） 内容仅左移一次包括CY 标志。



[描述示例]

ROLWC BC, 1; BC寄存器内容包括CY 标志左移一位。

6.9 位操作指令

如下是位操作指令。

MOV1

Move Single Bit
1位数据传送

[指令格式] MOV1 dst, src

[操作] dst ← src

[操作数]

助记符	操作数 （dst, src）
MOV1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit
	saddr.bit, CY

助记符	操作数 （dst, src）
MOV1	sfr.bit, CY
	A.bit, CY
	PSW.bit, CY
	[HL].bit, CY
	CY, ES:[HL].bit
	ES:[HL].bit, CY

[标志]

dst = CY

Z	AC	CY
		×

dst = PSW.bit

Z	AC	CY
×	×	

其它所有情况

Z	AC	CY

[描述]

- 由第二操作数指定的源操作数（src）的位数据传送到由第一操作数指定的目的操作数 （dst）。
- 当目的操作数 （dst）是 CY 或 PSW.bit时，仅相应的标志改变。

[描述示例]

MOV1 P3.4, CY; CY标志内容传送到端口3的第4位。

AND1

And Single Bit
1位数据逻辑与

[指令格式] AND1 dst, src

[操作] dst ← dst ∧ src

[操作数]

助记符	操作数 （dst, src）
AND1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit
	CY, ES:[HL].bit

[标志]

Z	AC	CY
		×

[描述]

- 由第一操作数指定的目的操作数 （dst）的位数据与由第二操作数指定的源操作数（src）的位数据逻辑与，得到的结果存储在目的操作数 （dst）中。
- 操作结果存储在CY标志中（由于目的操作数 （dst））。

[描述示例]

AND1 CY, FFE7FH.3; FFE7FH第3位与CY标志逻辑与的结果存储在CY标志中。

OR1

Or Single Bit
1位数据逻辑或

[指令格式] OR1 dst, src

[操作] dst ← dst ∨ src

[操作数]

助记符	操作数 （dst, src）
OR1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit
	CY, ES:[HL].bit

[标志]

Z	AC	CY
		×

[描述]

- 由第一操作数指定的目的操作数 （dst）的位数据与由第二操作数指定的源操作数（src）的位数据逻辑或，得到的结果存储在目的操作数 （dst）中。
- 操作结果存储在CY标志中（由于目的操作数 （dst））。

[描述示例]

OR1 CY, P2.5; 端口2第5位与CY标志逻辑或的结果存储在CY标志中。

XOR1

Exclusive Or Single Bit
1 位数据逻辑异或

[指令格式] XOR1 dst, src

[操作] dst ← dst ⫟ src

[操作数]

助记符	操作数 （dst, src）
XOR1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit
	CY, ES:[HL].bit

[标志]

Z	AC	CY
		×

[描述]

- 由第一操作数指定的目的操作数 （dst）的位数据与由第二操作数指定的源操作数（src）的位数据逻辑异或，得到的结果存储在目的操作数 （dst）中。
- 操作结果存储在CY标志中（由于目的操作数 （dst））。

[描述示例]

XOR1 CY, A.7; A寄存器第7位与CY标志逻辑异或的结果存储在CY标志中。

SET1

Set Single Bit(Carry Flag)
1 位数据置位

[指令格式] SET1 dst

[操作] dst ← 1

[操作数]

助记符	操作数 （dst）
SET1	saddr.bit
	sfr.bit
	A.bit
	!addr16.bit
	PSW.bit
	[HL].bit
	ES:!addr16.bit
	ES:[HL].bit
	CY

[标志]

dst = PSW.bit

Z	AC	CY
×	×	×

dst = CY

Z	AC	CY
		1

其它所有情况

Z	AC	CY

[描述]

- 目的操作数 （dst） 被设置为（1）。
- 当目的操作数 （dst）是 CY 或 PSW.bit，只有相应的标志设置为（1）。

[描述示例]

SET1 FFE55H.1; FFE55H的第一位设置为（1）。

CLR1

Clear Single Bit(Carry Flag)
1 位数据清零

[指令格式] CLR1 dst

[操作] dst ← 0

[操作数]

助记符	操作数 （dst）
CLR1	saddr.bit
	sfr.bit
	A.bit
	!addr16.bit
	PSW.bit
	[HL].bit
	ES:!addr16.bit
	ES:[HL].bit
	CY

[标志]

dst = PSW.bit

Z	AC	CY
×	×	×

dst = CY

Z	AC	CY
		0

其它所有情况

Z	AC	CY

[描述]

- 目的操作数 （dst） 被清零（0）。
- 当目的操作数 （dst）是 CY 或 PSW.bit，只有相应的标志被清零（0）。

[描述示例]

CLR1 P3.7; 端口3的第7位被清零（0）。

NOT1

Not Single Bit(Carry Flag)
1 位数据逻辑反

[指令格式] NOT1 dst

[操作] dst ← $\overline{\text{dst}}$

[操作数]

助记符	操作数 （dst）
NOT1	CY

[标志]

Z	AC	CY
		×

[描述]

- CY 标志反相。

[描述示例]

NOT1 CY; CY标志反相。

6.10 调用 / 返回指令

如下是调用 / 返回指令。

CALL

Call
子程序调用

[指令格式] CALL 目标地址

[操作]

(SP−2) ← (PC+n)S,
(SP−3) ← (PC+n)H,
(SP−4) ← (PC+n)L,
SP ← SP−4
PC ← 目标地址

备注 当使用!!addr20时n为4，当使用!addr16 或 $!addr20时n为3，当使用AX、BC、DE或 HL时n为2。

[操作数]

助记符	操作数 （target）
CALL	AX
	BC
	DE
	HL
	$!addr20
	!addr16
	!!addr20

[标志]

Z	AC	CY

[描述]

- 它是一个通过一个20/16位绝对地址或者寄存器间接寻址的子程序调用。
- 下一条指令的开始地址（PC + n） 保存在堆栈内，并且跳转到目标操作数 （目标地址）指定的地址处执行。

[描述示例]

CALL !!3E000H; 子程序调用地址3E000H

CALLT

Call Table
子程序调用（参考调用表）

[指令格式] CALLT [addr5]

[操作]

(SP−2) ← (PC+2)S,
(SP−3) ← (PC+2)H,
(SP−4) ← (PC+2)L,
PCS ← 0000,
<R> PCH ← (0000, addr5+1),
<R> PCL ← (0000, addr5)
SP ← SP−4

[操作数]

助记符	操作数 （[addr5]）
CALLT	[addr5]

[标志]

Z	AC	CY

[描述]

- 它是一个调用表参考的子程序调用。
- <R> 下一条指令的开始地址（PC+2）存于堆栈中，程序跳转到调用表的字地址（00080H到000BFH的偶地址，高4为固定为000B，低16位由addr5指定）。

[描述示例]

CALLT [80H]; 子程序调用，跳转到00080H和00081H指向的字数据地址处执行。

[备注]

仅可指定偶地址（不能指定奇地址）。

<R> addr5：从0080H到00BFH的立即数或标号（仅偶地址）
（0080H到00BFH的偶地址，第15到6位固定为0000000010B，第0位固定为0B，第5到1位变化）

BRK

Break
软件向量中断

[指令格式] BRK

[操作]

(SP–1) ← PSW,
(SP–2) ← (PC+2)S,
(SP–3) ← (PC+2)H,
(SP–4) ← (PC+2)L,
PCS ← 0000,
PCH ← (0007FH),
PCL ← (0007FH),
SP ← SP–4,
IE ← 0

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 它是一个软件中断指令。
- PSW和下条指令地址（PC+2）存到堆栈。然后，IE标志清零，程序跳转到矢量地址（0007EH，0007FH）字节数据指定的地址。
  由于IE标志清零，因此禁止可屏蔽向量中断。
- RETB指令用来从此指令产生的软件向量中断返回。

RET

Return
从子程序返回

[指令格式] RET

[操作]

$PC_L \leftarrow (SP)$,
$PC_H \leftarrow (SP+1)$,
$PC_S \leftarrow (SP+2)$
$SP \leftarrow SP+4$

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 它是一个返回指令，从CALL和CALLT指令的子程序调用返回。
- 堆栈中保存的字数据返回到PC，并且程序从子程序返回。

RETI

Return from Interrupt
从硬件向量中断返回

[指令格式] RETI

[操作]

PC_L ← (SP),
PC_H ← (SP+1),
PC_S ← (SP+2),
PSW ← (SP+3),
SP ← SP+4,

[操作数]

无

[标志]

Z	AC	CY
R	R	R

<R> **[描述]**

- 它是一个从向量中断返回的指令。
- 堆栈中保存的数据返回给PC 和 PSW，并且程序从中断服务子程序返回。
- 此指令不能用作从BRK指令产生的软件中断返回。
- 在该指令和下一条指令执行中间，不会响应任何中断。

[注意事项]

<R> 确认用RETI指令从不可屏蔽中断恢复。

RETB

Return from Break
从软件向量中断返回

[指令格式] RETB

[操作]

PC_L ← (SP),
PC_H ← (SP+1),
PC_S ← (SP+2),
PSW ← (SP+3),
SP ← SP+4

[操作数]

无

[标志]

Z	AC	CY
R	R	R

[描述]

- 它是一个从BRK指令产生的软件向量中断返回的指令。
- 堆栈中保存的数据返回给PC 和 PSW，并且程序从中断服务子程序返回。
- 在该指令和下一条指令执行中间，不会响应任何中断。

6.11 堆栈操作指令

如下是堆栈操作指令。

PUSH

Push
压栈

[指令格式] PUSH src

[操作]

当src = rp	当src = PSW
(SP−1) ← rp_H,	(SP−1) ← PSW
(SP−2) ← rp_L,	(SP−2) ← 00H
SP ← SP−2	SP ← SP−2

[操作数]

助记符	操作数 （src）
PUSH	PSW
	rp

[标志]

Z	AC	CY

[描述]

- 源操作数 （src） 指定的寄存器数据保存在堆栈中。

[描述示例]

PUSH AX; AX 寄存器内容保存在堆栈中。

POP

Pop
出栈

[指令格式] POP dst

[操作]

当 dst = rp	当 dst = PSW
$rp_L \leftarrow (SP)$,	PSW ← (SP+1)
$rp_H \leftarrow (SP+1)$,	SP ← SP+2
SP ← SP+2	

[操作数]

助记符	操作数 （dst）
POP	PSW
	rp

[标志]

dst = rp

Z	AC	CY

dst = PSW

Z	AC	CY
R	R	R

[描述]

- 数据从堆栈返回到目的操作数 （dst ）指定的寄存器中。
- 当操作数 是PSW时，每个标志都被堆栈数据更换。
- 在 POP PSW 指令和接下来的指令执行中，不会响应任何中断。

[描述示例]

POP AX; 堆栈数据返回到AX 寄存器中。

MOVW SP, src
MOVW AX, SP

Move Word
含有堆栈指针的字数据传输

[指令格式] MOVW dst, src

[操作] dst ← src

[操作数]

助记符	操作数 （dst, src）
MOVW	SP, #word
	SP, AX
	AX, SP
	HL, SP
	BC, SP
	DE, SP

[标志]

Z	AC	CY

[描述]

- 它是操作堆栈指针内容的指令。
- 第二个操作数指定的源操作数 （src） 保存在第一个操作数指定的目的操作数 （dst） 中。

[描述示例]

MOVW SP, #FE1FH; FE1FH保存在堆栈指针中。

ADDW SP, #byte

Add stack pointer
堆栈指针加

[指令格式] ADDW SP, src

[操作] SP ← SP+src

[操作数]

助记符	操作数 （src）
ADDW	SP, #byte

[标志]

Z	AC	CY

[描述]

- 第一个操作数指定的堆栈指针与第二个操作数指定的源操作数 （src）相加，结果保存在堆栈指针中。

[描述示例]

ADDW SP, #12H; 堆栈指针与12H相加，结果保存在堆栈指针中。

SUBW SP, #byte

Sub stack pointer
堆栈指针减

[指令格式] SUBW SP, src

[操作] SP ← SP–src

[操作数]

助记符	操作数 （src）
SUBW	SP, #byte

[标志]

Z	AC	CY

[描述]

- 第一个操作数指定的堆栈指针减第二个操作数指定的源操作数 （src），结果保存在堆栈指针中。

[描述示例]

SUBW SP, #12H; 从堆栈指针减12H，结果保存在堆栈指针中。

6.12 无条件跳转指令

如下是无条件跳转指令。

BR ... 161

BR

Branch
无条件跳转

[指令格式] BR 目标地址

[操作] PC ← 目标地址

[操作数]

助记符	操作数 （target）
BR	AX
	$addr20
	$!addr20
	!addr16
	!!addr20

[标志]

Z	AC	CY

[描述]

- 它是一条无条件跳转指令。
- 目的地址操作数 （目的地址）的字数据传输给PC，并且程序跳转。

[描述示例]

BR !!12345H; 跳转到地址12345H。

6.13 条件跳转指令

如下是条件跳转指令。

BC

Branch if Carry
带进位标志 （CY = 1）的条件跳转

[指令格式] BC $addr20

[操作] PC ← PC+2+jdisp8 如果 CY = 1

[操作数]

助记符	操作数 （$addr20）
BC	$addr20

[标志]

Z	AC	CY

[描述]

- 当CY = 1，程序跳转到操作数指定的地址处执行。
  当 CY = 0，不执行任何处理，并且执行下一条指令。

[描述示例]

BC $00300H; 当CY = 1，程序跳转到00300H （该指令的开始地址的地址范围从0027FH 到 0037EH）。

BNC

Branch if Not Carry
带进位标志 （CY = 0）的条件跳转

[指令格式] BNC $addr20

[操作] PC ← PC+2+jdisp8 如果 CY = 0

[操作数]

助记符	操作数 （$addr20）
BNC	$addr20

[标志]

Z	AC	CY

[描述]

- 当CY = 0，程序跳转到操作数指定的地址处执行。
  当 CY = 1，不执行任何处理，并且执行下一条指令。

[描述示例]

BNC $00300H; 当CY = 0，程序跳转到00300H （该指令的开始地址的地址范围从0027FH 到 0037EH）。

BZ

Branch if Zero
带零标志 （Z = 1）的条件跳转

[指令格式] BZ $addr20

[操作] PC ← PC+2+jdisp8 如果 Z = 1

[操作数]

助记符	操作数 （$addr20）
BZ	$addr20

[标志]

Z	AC	CY

[描述]

- 当Z = 1，程序跳转到操作数指定的地址处执行。
  当Z = 0, 不执行任何处理，并且执行下一条指令。

[描述示例]

DEC B

BZ $003C5H; 当B 寄存器是0，程序跳转到003C5H （该指令的开始地址的地址范围从00344H 到 00443H）。

BNZ

Branch if Not Zero
带零标志 （Z = 0）的条件跳转

[指令格式] BNZ $addr20

[操作] PC ← PC+2+jdisp8 如果 Z = 0

[操作数]

助记符	操作数 （$addr20）
BNZ	$addr20

[标志]

Z	AC	CY

[描述]

- 当Z = 0，程序跳转到操作数指定的地址处执行。
  当Z = 1, 不执行任何处理，并且执行下一条指令。

[描述示例]

CMP A, #55H
BNZ $00A39H; 如果A寄存器不是55H，程序跳转到00A39H（该指令的开始地址的地址范围从009B8H 到00AB7H）。

BH

Branch if Higher than
数值比较 （（Z ∨ CY） = 0）条件跳转

[指令格式] BH $addr20

[操作] PC ← PC+3+jdisp8 如果 （Z ∨ CY） = 0

[操作数]

助记符	操作数 （$addr20）
BH	$addr20

[标志]

Z	AC	CY

[描述]

- 当（Z ∨ CY） = 0, 跳转到由操作数指定的地址。
  当（Z ∨ CY） = 1, 不执行任何处理，并且执行下一条指令。
- 此指令用于判断哪个无符号数据值更大。检测此指令之前的CMP指令中的第一操作数是否大于第二操作数。

[描述示例]

```
CMP A, C
BH $00356H;
```

当A寄存器的内容大于C寄存器的内容跳转到地址00356H （该指令的开始地址的地址范围从002D4H到003D3H）。

BNH

Branch if Not Higher than
数值比较 （（Z ∨ CY） = 1）条件跳转

[指令格式] BNH $addr20

[操作] PC ← PC+3+jdisp8 如果 （Z ∨ CY） = 1

[操作数]

助记符	操作数 （$addr20）
BNH	$addr20

[标志]

Z	AC	CY

[描述]

- 当（Z ∨ CY） = 1, 跳转到由操作数指定的地址。
  当（Z ∨ CY） = 0, 不执行任何处理，并且执行下一条指令。
- 此指令用于判断哪个无符号数据值更大。检测此指令之前的CMP指令中的第一操作数是否不大于第二操作数（第一操作数等于或小于第二操作数）。

[描述示例]

```
CMP A, C
BNH $00356H;
```

当A寄存器的内容等于或小于C寄存器的内容跳转到地址00356H （该指令的开始地址的地址范围从002D4H到003D3H）。

BT

Branch if True
带位测试 （字节数据位 = 1）的条件跳转

[指令格式] BT bit, $addr20

[操作] PC ← PC+b+jdisp8 如果 bit = 1

[操作数]

助记符	操作数 （bit, $addr20）	b （字节数）
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

[标志]

Z	AC	CY

[描述]

- 如果第一个操作数 （位） 内容已被设置为（1），程序跳转到第二个操作数（$addr20）指定的地址处。
  如果第一个操作数 （位） 内容没被设置为（1），不执行任何操作然后执行下一条指令。

[描述示例]

BT FFE47H.3, $0055CH; 当地址FFE47H的第3位为1时，程序跳转到0055CH（该指令开始地址设置范围为004DAH到005D9H）。

BF

Branch if False
带位测试 （字节数据位 = 0）的条件跳转

[指令格式] BF bit, $addr20

[操作] PC ← PC+b+jdisp8 如果 bit = 0

[操作数]

助记符	操作数 （bit, $addr20）	b （字节数）
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

[标志]

Z	AC	CY

[描述]

- 如果第一个操作数 （位） 内容已被清零（0），程序跳转到第二个操作数（$addr20）指定的地址处。
  如果第一个操作数 （位） 内容没被清零（0），不执行任何操作然后执行下一条指令。

[描述示例]

BF P2.2, $01549H; 当端口2的第2位为0时，程序跳转到01549H（该指令开始地址设置范围为014C6H到015C5H）。

BTCLR

Branch if True and Clear
带位测试 （字节数据位 = 1）的条件跳转并清零

[指令格式] BTCLR bit, $addr20

[操作] PC ← PC+b+jdisp8 如果 bit = 1，bit ← 0

[操作数]

助记符	操作数 （bit, $addr20）	b （字节数）
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

[标志]

bit = PSW.bit

Z	AC	CY
×	×	×

其它所有情况

Z	AC	CY

[描述]

- 如果第一个操作数 （位） 内容已被设置为（1），则清零并且程序跳转到第二个操作数指定的地址处。
  如果第一个操作数 （位） 内容没被设置为（1），不执行任何操作然后执行下一条指令。
- 当第一个操作数 （位） 为PSW.bit时，相应标志内容清零。

[描述示例]

BTCLR PSW.0, $00356H; 当PSW的第0位（CY标志）为1时，CY标志清零，程序跳转到00356H（该指令开始地址设置范围为002D4H到003D3H）。

6.14 条件跳过指令

如下指令是条件跳过指令。

SKC

Skip if CY
带进位标志（CY = 1）跳过

[指令格式] SKC

[操作] 如果CY = 1跳过下一条指令

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 当CY = 1, 跳过下一条指令。接着执行NOP指令，执行时间为1个时钟。但是，如果下一条指令是前缀指令（以“ES：”表示）, 执行时间为2个时钟。
- 当CY = 0, 执行下一条指令。

[描述示例]

MOV A, #55H
SKC
ADD A, #55H; 当CY = 1时A寄存器的值= AAH, 当CY = 1时A寄存器的值= 55H。

SKNC

Skip if not CY
带进位标志（CY = 0）跳过

[指令格式] SKNC

[操作] 如果CY = 0跳过下一条指令

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 当CY = 0，跳过下一条指令。接着执行NOP指令，执行时间为1个时钟。但是，如果下一条指令是前缀指令（以“ES：”表示），执行时间为2个时钟。
- 当CY = 1，执行下一条指令。

[描述示例]

MOV A, #55H
SKNC
ADD A, #55H; 当CY = 1时A寄存器的值= AAH，当CY = 0时A寄存器的值= 55H。

SKZ

Skip if Z
零标志 （Z = 1）跳过

[指令格式] SKZ

[操作] 如果Z = 1跳过下一条指令

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 当Z = 1，跳过下一条指令。接着执行NOP指令，执行时间为1个时钟。但是，如果下一条指令是前缀指令（以“ES：”表示），执行时间为2个时钟。
- 当Z = 0，执行下一条指令。

[描述示例]

MOV A, #55H
SKZ
ADD A, #55H; 当Z =0时A寄存器的值= AAH，当Z = 1时A寄存器的值= 55H。

SKNZ

Skip if not Z
零标志 （Z = 0）跳过

[指令格式] SKNZ

[操作] 如果Z = 0跳过下一条指令

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 当Z = 0，跳过下一条指令。接着执行NOP指令，执行时间为1个时钟。但是，如果下一条指令是前缀指令（以“ES：”表示），执行时间为2个时钟。
- 当Z = 1，执行下一条指令。

[描述示例]

MOV A, #55H
SKNZ
ADD A, #55H; 当Z = 1时A寄存器的值= AAH，当Z = 0时A寄存器的值= 55H。

SKH

Skip if Higher than
数值比较（（Z ∨ CY）= 0）跳过

[指令格式] SKH

[操作] 如果（Z ∨ CY）= 0 跳过下一条指令

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 当（Z ∨ CY）= 0，跳过下一条指令。接着执行NOP指令，执行时间为1个时钟。但是，如果下一条指令是前缀指令（以“ES：”表示），执行时间为2个时钟。
- 当（Z ∨ CY）= 1，执行下一条指令。

[描述示例]

```
CMP A, #80H
SKH
CALL !!TARGET;
```

当A 寄存器的内容大于80H，跳过CALL 指令并执行下一条指令。
当A 寄存器的内容为80H 或更小， 执行下一条CALL 指令并跳转到目标地址。

SKNH

Skip if not Higher than
数值比较（（Z ∨ CY） = 1）跳过

[指令格式] SKNH

[操作] 如果（Z ∨ CY） = 1 跳过下一条指令

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 当（Z ∨ CY） = 1，跳过下一条指令。接着执行NOP，指令执行时间为1个时钟。但是，如果下一条指令是前缀指令（以“ES：”表示），执行时间为2个时钟。
- 当（Z ∨ CY） = 0，执行下一条指令。

[描述示例]

```
CMP A, #80H
SKNH
CALL !!TARGET;
```

当A 寄存器的内容为80H 或更小，跳过CALL 指令并执行下一条指令。
当A 寄存器的内容大于80H ，执行下一条CALL 指令并跳转到目标地址。

6.15 CPU 控制指令

如下指令是CPU 控制指令。

SEL RBn

Select Register Bank
寄存器Bank 选择

[指令格式] SEL RBn

[操作] RBS0, RBS1 ← n; （n = 0 到 3）

[操作数]

助记符	操作数 （RBn）
SEL	RBn

[标志]

Z	AC	CY

[描述]

- 通过操作数（RBn）指定下一条及后面的指令所用的寄存器bank。
- RBn的范围从RB0到RB3。

[描述示例]

SEL RB2; 选择寄存器bank2作为下一条及后面的指令所用的寄存器bank。

NOP

No operation
无操作

[指令格式] NOP

[操作] 无操作

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 不执行任何操作，只消耗时间。

EI

Enable Interrupt
中断允许

[指令格式] EI

[操作] IE ← 1

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 可屏蔽的中断响应允许状态被设置（通过设置中断允许标志 （IE）（1））。
- 在此指令与下一条执行之间不响应中断。
- 如果执行该指令，其它源的向量中断响应可被禁止。如需了解详细信息参见各产品用户手册的中断功能。

DI

Disable Interrupt
中断禁止

[指令格式] DI

[操作] IE ← 0

[操作数]

无

[标志]

Z	AC	CY

[描述]

- 带有向量中断的可屏蔽的中断响应被禁止（通过清零中断允许标志 （IE） （0））。
- 在此指令与下一条执行之间不响应中断。
- 如需了解详细信息参见各产品用户手册的中断功能。

HALT

Halt
HALT 模式设置

[指令格式] HALT

[操作] 设置HALT 模式

[操作数]
无

[标志]

Z	AC	CY

[描述]
- 该指令用来设置HALT模式，停止CPU操作时钟。和正常操作模式一起间歇性的工作可以降低系统的整体功耗。

STOP

Stop
Stop 模式设置

[指令格式] STOP

[操作] 设置 STOP 模式

[操作数]

无

[标志]

Z	AC	CY

[描述]

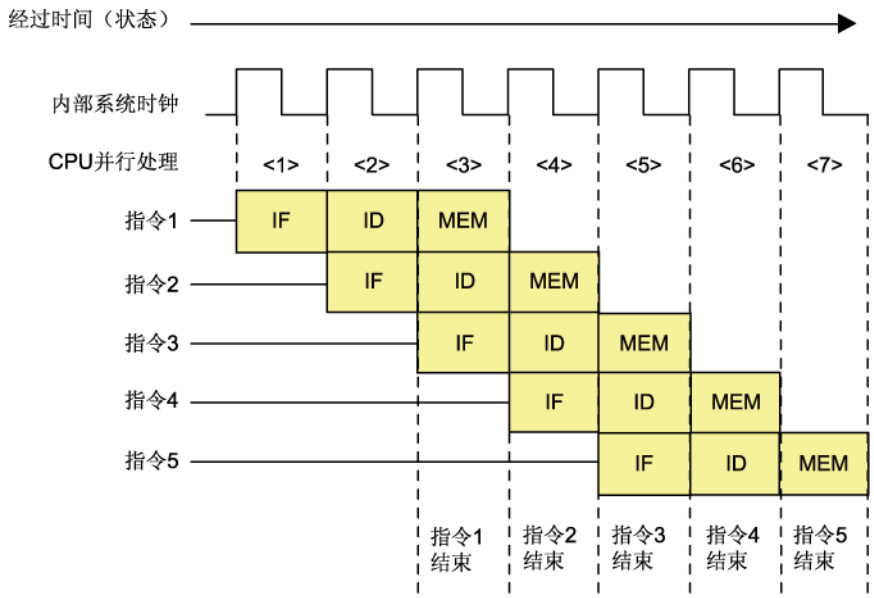
- 该指令用来设置STOP 模式停止主系统时钟振荡器，并且停止整个系统。功耗会被最小化到只有漏电流的情况。

第七章　流水线

7.1　特征

78K0R 微控制器使用三级流水线控制可使几乎所有指令单周期执行。指令分三个阶段执行：取指令（IF）、指令解码（ID）和访问存储器（MEM）。

图 7-1. 5 个指令的流水线执行（举例）



- IF （取指令）：　　取指令并增加指针。
- ID （指令解码）：　　指令解码并计算地址。
- MEM （访问存储器）：　执行解码的指令并访问存储器的目标地址。

7.2 操作时钟数

尽管在其它一些具有流水线的微控制器中存在计数时钟无法计数的问题，但是 78K0R 微控制器通过在相同时钟数保持操作解决了这个问题，因此可以稳定的执行程序。

这些时钟数如表 **5.5 操作列表** 所示。

7.2.1 访问flash存储器的内容为数据

当访问 flash 存储器的内容为数据时，在 MEM 阶段停止流水线操作。因此，列表中的操作时钟数增加。关于详细情况，请参考 **5.5 操作列表**。

7.2.2 访问外部存储器的内容为数据

当访问外部存储器的内容为数据时， CPU 设置为等待模式。因此，列表中的操作时钟数增加。

关于增加的时钟数，请参考 **表 7-1** 。

表 7-1. 从/到外部存储器读 / 写期间的 CPU 等待时钟数

选择外部扩展时钟输出 （CLKOUT）	等待时钟数
f_{CLK}	3 个时钟
$f_{CLK}/2$	5 或 6 个时钟
$f_{CLK}/3$	7 到 9 个时钟
$f_{CLK}/4$	9 到 12 个时钟

7.2.3 从RAM取指令

当从 RAM 取数据时，因为从 RAM 读有延迟，指令队列为空。因此 CPU 等待直到数据设置到指令队列。在从 RAM 读取期间，如果访问 RAM，CPU 也等待。

7.2.4 从外部存储器取指令

当从外部存储器取数据时，因为从外部存储器读有延迟，指令队列为空。因此 CPU 等待直到数据设置到指令队列。在从外部存储器读取期间，如果访问外部存储器，CPU 也等待。

关于增加的时钟数，请参考 **表 7-2** 。

表 7-2. 当从外部存储器读取数据时的 CPU 等待时钟数

选择外部扩展时钟输出 （CLKOUT）	等待时钟数
f_{CLK}	3 个时钟
$f_{CLK}/2$	5 或 6 个时钟
$f_{CLK}/3$	7 到 9 个时钟
$f_{CLK}/4$	9 到 12 个时钟

7.2.5 关于组合指令的问题

如果在写入用于间接访问的寄存器之后立即间接访问寄存器的数据，将会插入一个时钟的等待。

寄存器名称	前一条指令	下一条指令操作数 （或指令）
DE	对 D 寄存器[注] 的写指令 对 E 寄存器[注]的写指令 对 DE 寄存器[注]的写指令 SEL RBn	[DE], [DE+byte]
HL	对 H 寄存器[注]的写指令 对 L 寄存器[注]的写指令 对 HL 寄存器[注]的写指令 SEL RBn	[HL], [HL+byte], [HL+B], [HL+C], [HL].bit
B	对 B 寄存器[注]的写指令 SEL RBn	Word[B], [HL+B]
C	对 C 寄存器[注]的写指令 SEL RBn	Word[C], [HL+C]
BC	对 B 寄存器[注]的写指令 对 C 寄存器[注]的写指令 对 BC 寄存器[注]的写指令 SEL RBn	Word[BC], [HL+B], [HL+C]

SP	MOVW SP, #word
	MOVW SP, AX
	ADDW SP, #byte
	SUBW SP, #byte
		[SP+byte]
		CALL 指令，CALLT 指令，BRK 指令，SOFT 指令，RET 指令，RETI 指令，RETB 指令，中断，PUSH 指令，POP 指令

CS	MOV CS, #byte MOV CS, A	CALL rp BR AX
AX	对 A 寄存器[注]的写指令 对 X 寄存器[注]的写指令 对 AX 寄存器[注]的写指令 SEL RBn	BR AX
AX BC DE HL	对 A 寄存器[注]的写指令 对 X 寄存器[注]的写指令 对 B 寄存器[注]的写指令 对 C 寄存器[注]的写指令 对 D 寄存器[注]的写指令 对 E 寄存器[注]的写指令 对 H 寄存器[注]的写指令 对 L 寄存器[注]的写指令 对 AX 寄存器[注]的写指令 对 BC 寄存器[注]的写指令 对 DE 寄存器[注]的写指令 对 HL 寄存器[注]的写指令 SEL RBn	CALL rp

注 在直接寻址，短直接寻址，寄存器间接寻址，基址寻址，或基址变址寻址期间，当覆写目标寄存器的值时，寄存器写指令也需要插入等待。

附录 A 指令索引（助记符：按功能）

[无条件转移指令]

[条件转移指令]

[条件跳转指令]

[CPU 控制指令]

附录 B 指令索引（助记符：按字母顺序）

[S]

[X]

C.1 此版本的主要修改记录

页码	内容	类别
第二章 存储器空间		
p.12.	增加了 **2.4 特殊功能寄存器（SFR）区域**的描述	（c）
p.13	增加了 **2.5 扩展 SFR（第二 SFR）区域**的描述	（c）
第五章 指令集		
p.35	在**表 5-2. 操作栏**中的符号中增加了 addr5	（c）
pp.37 到 53	在**表 5-5 操作列表** 中修改 **备注 1**	（a）
p.51	改变了 CALLT 指令的操作	（b）
第六章 指令说明		
p.149	改变了 CALLT 指令中[**操作**]，[**描述**]和[**备注**]	（b，c）
p.152	改变了 RETI 指令的[**描述**]和[**注意事项**]	（b）

备注 上表中的“类别”如下所示。

（a）：错误改正，（b）：增加/改变说明，（c）：增加/改变描述或注，（d）：增加/改变封面、编号或管理部门，（e）：增加 / 改变相关文档

C.2 以前版本的修改记录

以前版本的修改记录显示如下。章节表示各个版本的章节。

版本	内容	章节
第二版	增加 **2.6 外部存储器空间 注意事项**	第二章 存储器空间
	在 **4.2.4 短直接寻址**中增加描述方法和**备注**	第四章 寻址
	在 **表 5-5 操作列表 中增加备注**	第五章 指令集
	在 **表 5-5 操作列表**中增加操作数 ADDW、SUBW、CMPW、INC、DEC、INCW 和 DECW	
	在 **表 5-5 操作列表**中增加条件转移指令 BH 和 BNH	
	在 **表 5-5 操作列表**中增加条件跳转指令 SKH 和 SKNH	
	在 **表 5-6 指令格式列表**中增加 MOV、ADDW、SUBW、CMPW、INC、DEC、INCW 和 DECW 操作数	
	在 **表 5-6 指令格式列表** 中增加 BH、BNH、SKH 和 SKNH 指令	
	在 **表 5-8 指令映射图 （2nd 映射图）**中增加 ADDW、SUBW、CMPW、INC、DEC、INCW 和 DECW 操作数	
	在 **第六章 指令说明** 中增加 MOV、ADDW、SUBW、CMPW、INC、INCW 和 DECW 操作数	第六章 指令说明
	在 **6.13 条件转移指令** 中增加 BH 和 BNH 指令	
	在 **6.14 条件跳转指令** 中增加 SKH 和 SKNH 指令	
	增加 **附录 C 修改记录**	附录 C 修改记录
第三版	在 **2.2.1 镜像区域** 中修改**示例 1**, **2**	第二章 存储器空间
	在**表 5-5 操作列表** 中修改 **注**	第五章 指令集
	在 **表 5-5 操作列表** 中修改 CALL 和 BR 指令的操作数描述	
	在**第六章 指令说明** 中增加 DEC 指令的操作数	第六章 指令说明
	在 **第六章 指令说明** 中修改 CALL 和 BR 指令的操作数描述	
	在 **第六章 指令说明** 中修改 BR 指令的 **[指令格式]**	

[备忘录]

详细信息请联系：

中国区

MCU 技术支持热线：
电话：+86-400-700-0606 (普通话)
服务时间：9:00-12:00，13:00-17:00 （不含法定节假日）

网址：
http://www.cn.necel.com/ （中文）
http://www.necel.com/ （英文）

[北京]

日电电子（中国）有限公司
中国北京市海淀区知春路 27 号
量子芯座 7，8，9，15 层
电话：（+86）10-8235-1155
传真：（+86）10-8235-7679

[上海]

日电电子（中国）有限公司上海分公司
中国上海市浦东新区银城中路 200 号
中银大厦 2409-2412 和 2509-2510 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

上海恩益禧电子国际贸易有限公司
中国上海市浦东新区银城中路 200 号
中银大厦 2511-2512 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[深圳]

日电电子（中国）有限公司深圳分公司
深圳市福田区益田路卓越时代广场大厦 39 楼
3901，3902，3909 室
电话：（+86）755-8282-9800
传真：（+86）755-8282-9899

[香港]

香港日电电子有限公司
香港九龙旺角太子道西 193 号新世纪广场
第 2 座 16 楼 1601-1613 室
电话： （+852）2886-9318
传真： （+852）2886-9022
2886-9044

[成都]

日电电子（中国）有限公司成都分公司
成都市二环路南三段 15 号天华大厦 7 楼 703 室
电话：(+86)28-8512-5224
传真：(+86)28-8512-5334