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

78K/0S 系列

8 位单片微控制器

指令

适用于 78K/0S 系列

文档编号. U11047CA3V0UMJ1 (第三版)
发行日期 2007 年 6 月 N CP(K)

[备忘录]

CMOS设备的注释

① 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。对于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。

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在高电平或低电平。每一个未使用引脚都应该通过附加电阻连接到VDD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 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设备并没有被初始化。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。

EEPROM是NEC Electronics Corporation的商标。

这些产品从日本出口是由日本政府调节的。一些或者所有产品如果没有政府的授权是不允许出口的。从其他国家再出口一些或者所有这些产品同样也是要那个国家授权的。如有问题请联系 NEC 销售代理。

如下产品的关于 EEPROM 微控制器专利生产和销售是基于 CP8 Transac 的许可。
这些产品不能用作 IC 卡 (SMART CARD)。

适用的产品: μPD789146, 789156, 789197AY, 789217AY 子系列

订购 NEC I^2C 产品转让了 Philips I^2C 专利权下的许可来用于 I^2C 系统中，假如系统遵从 Philips 定义的 I^2C 标准规范。

适用的产品： μPD789197AY, 789217AY 子系列

- 本文档信息于 1999 年 3 月开始使用。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。本文件所登载内容的错误，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或为日本电气电子株式会社（定义如上）开发或制造的产品。

M8E 00.4

详细信息请联系：

（中国区）

网址：
http://www.cn.necel.com/
http://www.necel.com/

[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

该版本主要修改点

页数	内容
整篇	• 增加如下目标产品 μPD789046, 789104, 789114, 789124, 789134, 789146, 789156, 789167, 789177, 789197AY, 789217AY, 789407A, 789417A 和789842 子系列
	• 删除如下目标产品。 μPD789407, 789417, and 789806Y Subseries
p. 52	修改 MOV PSW, #byte 指令码
p. 52	修改 MOVW rp, AX 指令码
p. 54	修改 XOR A, r 指令码
p. 54	修改 CMP A, r 指令码

标记 ☛ 显示主要修改点。

引言

读者对象

本手册适用于那些希望了解 78K0S/KA1+ 产品功能，并设计开发相关应用系统和程序的用户。

78K/0S 系列产品

- μPD789014 子系列：　μPD789011, 789012, 78P9014
- μPD789026 子系列：　μPD789022, 789024, 789025, 789026, 78F9026
- μPD789046 子系列[注]：　μPD789046, 78F9046
- μPD789104 子系列：　μPD789101, 789102, 789104
- μPD789114 子系列：　μPD789111, 789112, 789114, 78F9116
- μPD789124 子系列[注]：　μPD789121, 789122, 789124
- μPD789134 子系列[注]：　μPD789131, 789132, 789134, 78F9136
- μPD789146 子系列[注]：　μPD789144, 789146
- μPD789156 子系列[注]：　μPD789154, 789156, 78F9156
- μPD789167 子系列[注]：　μPD789166, 789167
- μPD789177 子系列[注]：　μPD789176, 789177, 78F9177
- μPD789197AY 子系列[注]：　μPD789196AY, 789197AY, 78F9197AY
- μPD789217AY 子系列[注]：　μPD789216AY, 789217AY, 78F9217AY
- μPD789407A 子系列：　μPD789405A, 789406A, 789407A
- μPD789417A 子系列：　μPD789415A, 789416A, 789417A, 78F9418A
- μPD789800 子系列：　μPD789800, 78F9801
- μPD789842 子系列[注]：　μPD789841, 789842, 78F9842

注　开发中

目的

本手册用于帮助用户了解下面组织中描述的功能。

组织

本手册主要分为以下部分：

- **CPU 功能**
- **指令集**
- **指令描述**

手册使用方法

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 指令索引。**
- 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**第四章 指令集**中的助记符，然后是 **第五章 指令描述**中的功能。
- 如何从整体上理解 78K/0S 系列产品指令的所有功能：
  → 根据**内容**顺序阅读手册。
- 如何获悉 78K/0S 系列产品的硬件功能：
  → 参考每个产品的用户手册 (请参见 **相关文档**)。

规定

数据规则:	数据的高位部分在左边，低位部分在右边
低电平有效表示法:	$\overline{\times\times\times}$（在引脚和信号名称上划一条线）
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 ... ×××× 或 ××××B
	十进制 ... ××××
	十六进制 ... ××××H

相关文档

本手册中指出的相关文档包括了最初的版本，但未注明。

○ **适用于 78K/0S 系列的文档**

文档名称	文档编号	
	英文	日文
用户手册指令	本手册	U11047J

○ **独立文档**

● ***μ*PD789014 子系列**

文档名称	文档编号	
	英文	日文
*μ*PD789011, 789012 数据手册	U11095E	U11095J
*μ*PD78P9014 数据手册	U10912E	U10912J
*μ*PD789014 子系列用户手册	U11187E	U11187J

● ***μ*PD789026 子系列**

文档名称	文档编号	
	英文	日文
*μ*PD789022, 789024, 789025, 789026 数据手册	U11715E	U11715J
*μ*PD78F9026 数据手册	U11858E	U11858J
*μ*PD789026 子系列用户手册	U11919E	U11919J

● ***μ*PD789046 子系列**

文档名称	文档编号	
	英文	日文
*μ*PD789046 初级产品信息	U13380E	U13380J
*μ*PD78F9046 初级产品信息	U13546E	U13546J
*μ*PD789046 子系列用户手册	U13600E	U13600J

- **μPD789104 子系列**

文档名称	文档编号	
	英文	英文
μPD789101, 789102, 789104 数据手册	准备中	U12815J
μPD789134 子系列用户手册	U13045E	U13045J

- **μPD789114 子系列**

文档名称	文档编号	
	英文	英文
μPD789111, 789112, 789114 初级产品信息	U13013E	U13013J
μPD78F9116 初级产品信息	U13037E	U13037J
μPD789134 子系列用户手册	U13045E	U13045J

- **μPD789124 子系列**

文档名称	文档编号	
	英文	英文
μPD789121, 789122, 789124 初级产品信息	U13025E	U13025J
μPD789134 子系列用户手册	U13045E	U13045J

- **μPD789134 子系列**

文档名称	文档编号	
	英文	英文
μPD789131, 789132, 789134 初级产品信息	U13015E	U13015J
μPD78F9136 初级产品信息	U13036E	U13036J
μPD789134 子系列用户手册	U13045E	U13045J

- **μPD789146, 789156 子系列**

文档名称	文档编号	
	英文	英文
μPD789144, 789146, 789154, 789156 初级产品信息	U13478E	U13478J
μPD78F9156 初级产品信息	准备中	U13756J
μPD789146, 789156 子系列用户手册	U13651E	U13651J

- **μPD789167, 789177 子系列**

文档名称	文档编号	
	英文	英文
μPD789166, 789167, 789176, 789177 初级产品信息	准备中	U14017J
μPD78F9177 初级产品信息	准备中	U14022J
μPD789177 子系列用户手册	准备中	准备中

• **μPD789197AY 子系列**

文档名称	文档编号	
	英文	英文
μPD789196AY, 789197AY 初级产品信息	U13853E	U13853J
μPD78F9197Y 初级产品信息	U13224E	U13224J
μPD789217Y 子系列用户手册	U13186E	U13186J

• **μPD789217AY 子系列**

文档名称	文档编号	
	英文	英文
μPD789216Y, 789217Y 初级产品信息	U13196E	U13196J
μPD78F9217Y 初级产品信息	U13205E	U13205J
μPD789217Y 子系列用户手册	U13186E	U13186J

• **μPD789407A, 789417A 子系列**

文档名称	文档编号	
	英文	英文
μPD789405A, 789406A, 789407A, 789415A, 789416A, 789417A Data Sheet	准备中	U14024J
μPD78F9418A 数据手册	准备中	准备中
μPD789407A, 789417A 子系列用户手册	准备中	U13952J

• **μPD789800 子系列**

文档名称	文档编号	
	英文	英文
μPD789800 数据手册	U12627E	U12627J
μPD78F9801 初级产品信息	U12626E	U12626J
μPD789800 子系列用户手册	U12978E	U12978J

• **μPD789842 子系列**

文档名称	文档编号	
	英文	英文
μPD789841, 789842 初级产品信息	U13790E	U13790J
μPD78F9842 初级产品信息	U13901E	U13901J
μPD789842 子系列用户手册	U13776E	U13776J

注意事项　如下文档可以在没有事先声明情况下更改。切记要使用在开始设计时使用最新版本文档。

目录

附图

附表

[备忘录]

第一章 存储器空间

1.1 存储器空间

78K/0S 系列产品程序存储器映射图根据内部存储器容量而不同。关于存储器映射地址区域的详细情况，请参考具体产品的用户手册。

1.2 内部程序存储器(内部 ROM)空间

78K/0S 系列产品具有下列地址空间的内部 ROM。程序和表数据等存储在 ROM 中。此存储器空间通常由程序计数器(PC)寻址。

表 1-1. 78K/0S 系列产品的内部 ROM 空间(1/2)

容量 / 地址空间 / 子系列名称	2 K 字节 0000H 到 07FFH	4 K 字节 0000H 到 0FFFH	8 K 字节 0000H 到 1FFFH	12 K 字节 0000H 到 2FFFH	16 K 字节 0000H 到 3FFFH	24 K 字节 0000H 到 5FFFH	32 K 字节 0000H 到 7FFFH
μPD789014 子系列	μPD789011	μPD789012	μPD78P9014				
μPD789026 子系列		μPD789022	μPD789024	μPD789025	μPD789026 μPD78F9026		
μPD789046 子系列					μPD789046 μPD78F9046		
μPD789104 子系列	μPD789101	μPD789102	μPD789104				
μPD789114 子系列	μPD789111	μPD789112	μPD789114		μPD78F9116		
μPD789124 子系列	μPD789121	μPD789122	μPD789124				
μPD789134 子系列	μPD789131	μPD789132	μPD789134		μPD78F9136		
μPD789146 子系列			μPD789144		μPD789146		
μPD789156 子系列			μPD789154		μPD789156 μPD78F9156		
μPD789167 子系列					μPD789166	μPD789167	
μPD789177 子系列					μPD789176	μPD789177 μPD78F9177	
μPD789197AY 子系列					μPD789196AY	μPD789197AY μPD78F9197AY	

表 1-1. 78K/0S 系列产品的内部 ROM 空间(2/2)

容量 / 地址空间 / 子系列名称	2 K 字节 0000H 到 07FFH	4 K 字节 0000H 到 0FFFH	8 K 字节 0000H 到 1FFFH	12 K 字节 0000H 到 2FFFH	16 K 字节 0000H 到 3FFFH	24 K 字节 0000H 到 5FFFH	32 K 字节 0000H 到 7FFFH
μPD789217AY 子系列					μPD789216AY	μPD789217AY μPD78F9217AY	
μPD789407A 子系列				μPD789405A	μPD789406A	μPD789407A	
μPD789417A 子系列				μPD789415A	μPD789416A	μPD789417A	μPD78F9418A
μPD789800 子系列			μPD789800		μPD78F9801		
μPD789842 子系列			μPD789841		μPD789842 μPD78F9842		

1.3 矢量表区域

矢量表区域存储当输入 $\overline{\text{RESET}}$ 信号或产生中断请求时程序跳转到的开始地址。16 位地址的低 8 位存储在偶地址中，高 8 位存储在奇地址中。

表 1-2. 矢量表(0000H 到 0013H) (μPD789014 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	000CH	INTSR/INTCSI0
0004H	INTWDT	000EH	INTST
0006H	INTP0	0010H	INTTM0
0008H	INTP1	0012H	INTTM1
000AH	INTP2		

表 1-3. 矢量表(0000H 到 002BH) (μPD789026 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	000CH	INTSR/INTCSI0
0004H	INTWDT	000EH	INTST
0006H	INTP0	0010H	INTTM0
0008H	INTP1	0014H	INTTM2
000AH	INTP2	002AH	INTKR

表 1-4. 矢量表(0000H 到 0019H) (μPD789046 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	000EH	INTST20
0004H	INTWDT	0010H	INTWT
0006H	INTP0	0012H	INTWTI
0008H	INTP1	0014H	INTTM80
000AH	INTP2	0016H	INTTM90
000CH	INTSR20/INTCSI20	0018H	INTKR00

表 1-5. 矢量表(0000H 到 0015H) (μPD789104, 789114, 789124, 789134 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	000CH	INTSR20/INTCSI20
0004H	INTWDT	000EH	INTST20
0006H	INTP0	0010H	INTTM80
0008H	INTP1	0012H	INTTM20
000AH	INTP2	0014H	INTAD0

表 1-6. 矢量表(0000H 到 0019H) (μPD789146, 789156 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	000EH	INTST20
0004H	INTWDT	0010H	INTTM80
0006H	INTP0	0012H	INTTM20
0008H	INTP1	0014H	INTAD0
000AH	INTP2	0016H	INTLVI0
000CH	INTSR20/INTCSI20	0018H	INTEE1

表 1-7. 矢量表(0000H 到 0023H) (μPD789167, 789177 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	0012H	INTWT
0004H	INTWDT	0014H	INTWTI
0006H	INTP0	0016H	INTTM80
0008H	INTP1	0018H	INTTM81
000AH	INTP2	001AH	INTTM82
000CH	INTP3	001CH	INTTM90
000EH	INTSR20/INTCSI20	0022H	INTAD0
0010H	INTST20		

表 1-8. 矢量表(0000H 到 0027H) (μPD789197AY, 789217AY 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	0016H	INTTM80
0004H	INTWDT	0018H	INTTM81
0006H	INTP0	001AH	INTTM82
0008H	INTP1	001CH	INTTM90
000AH	INTP2	001EH	INTSMB0
000CH	INTP3	0020H	INTSMBOV0
000EH	INTSR20/INTCSI20	0022H	INTAD0
0010H	INTST20	0024H	INTLVI0
0012H	INTWT	0026H	INTEE1
0014H	INTWTI		

表 1-9. 矢量表(0000H 到 0023H) (μPD789407A 和 μPD789417A 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	0014H	INTWTI
0004H	INTWDT	0016H	INTTM00
0006H	INTP0	0018H	INTTM01
0008H	INTP1	001AH	INTTM02
000AH	INTP2	001CH	INTTM50
000CH	INTP3	001EH	INTKR00
000EH	INTSR00/INTCSI00	0020H	INTAD0
0010H	INTST00	0022H	INTCMP0
0012H	INTWT		

表 1-10. 矢量表(0000H 到 0019H) (μPD789800 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	000EH	INTUSBRE
0004H	INTWDT	0010H	INTP0
0006H	INTUSBTM	0012H	INTCSI10
0008H	INTUSBRT	0014H	INTTM00
000AH	INTUSBRD	0016H	INTTM01
000CH	INTUSBST	0018H	INTKR00

表 1-11. 矢量表(0000H 到 0023H) (μPD789842 子系列)

矢量表地址	中断请求	矢量表地址	中断请求
0000H	$\overline{\text{RESET}}$ 输入	0016H	INTST00
0004H	INTWDT	0018H	INTWT
0006H	INTP0	001AH	INTWTI
0008H	INTP1	001CH	INTTM80
000AH	INTTM7	001EH	INTTM81
000CH	INTSER00	0020H	INTTM82
000EH	INTSR00	0022H	INTAD

1.4 CALLT 指令表区域

在 64 字节地址区域 0040H 到 007FH 中，可以存储 1 字节调用指令(CALLT)的子程序入口地址。

1.5 内部数据存储器空间

78K/0S 系列产品包括如下数据存储器：

(1) 内部高速 RAM

78K/0S 系列产品包括表 1-12 中所示地址空间中的内部高速 RAM。

内部高速 RAM 也可用作堆栈存储器。

(2) LCD 显示 RAM (μPD789407A 和 μPD789417A 子系列)

LCD 显示 RAM 分配在 FA00H 与 FA1BH 之间的区域。

LCD 显示 RAM 也可用作普通 RAM。

(3) EEPROM™ (μPD789146, 789156, 789197AY, 789217AY 子系列)

电可擦除 PROM (EEPROM)分配在表 1-12 中所示的地址空间中。

不象普通 RAM，即使关闭电源 EEPROM 也能保持数据。而且，不象 EPROM，EEPROM 的内容可电擦除，而不需要用紫外线照射芯片。

表 1-12. 78K/0S 系列产品的内部数据存储器空间(1/2)

子系列名称	产品名称	高速 RAM	LCD 显示 RAM	EEPROM
μPD789014 子系列	μPD789011	FE80H 到 FEFFH (128 字节)	—	—
	μPD789012			
	μPD78P9014	FE00H 到 FEFFH (256 字节)		
μPD789026 子系列	μPD789022	FE00H 到 FEFFH (256 字节)	—	—
	μPD789024			
	μPD789025	FD00H 到 FEFFH (512 字节)		
	μPD789026			
	μPD78F9026			
μPD789046 子系列	μPD789046	FD00H 到 FEFFH (512 字节)	—	—
	μPD78F9046			
μPD789104 子系列	μPD789101	FE00H 到 FEFFH (256 字节)	—	—
	μPD789102			
	μPD789104			
μPD789114 子系列	μPD789111	FE00H 到 FEFFH (256 字节)	—	—
	μPD789112			
	μPD789114			
	μPD78F9116			

表 1-12. 78K/0S 系列产品的内部数据存储器空间(2/2)

子系列名称	产品名称	高速 RAM	LCD 显示 RAM	EEPROM
μPD789124 子系列	μPD789121	FE00H 到 FEFFH (256 字节)	—	—
	μPD789122			
	μPD789124			
μPD789134 子系列	μPD789131	FE00H 到 FEFFH (256 字节)	—	—
	μPD789132			
	μPD789134			
	μPD78F9136			
μPD789146 子系列	μPD789144	FE00H 到 FEFFH (256 字节)	—	F800H 到 F8FFH (256 字节)
	μPD789146			
μPD789156 子系列	μPD789154	FE00H 到 FEFFH (256 字节)	—	F800H 到 F8FFH (256 字节)
	μPD789156			
	μPD78F9156			
μPD789167 子系列	μPD789166	FD00H 到 FEFFH (512 字节)	—	—
	μPD789167			
μPD789177 子系列	μPD789176	FD00H 到 FEFFH (512 字节)	—	—
	μPD789177			
	μPD78F9177			
μPD789197AY 子系列	μPD789196AY	FD00H 到 FEFFH (512 字节)	—	F800H 到 F87FH (128 字节)
	μPD789197AY			
	μPD78F9197AY			
μPD789217AY 子系列	μPD789216AY	FD00H 到 FEFFH (512 字节)	—	F800H 到 F87FH (128 字节)
	μPD789217AY			
	μPD78F9217AY			
μPD789407A 子系列	μPD789405A	FD00H 到 FEFFH (512 字节)	FA00H 到 FA1BH (28 字节)	—
	μPD789406A			
	μPD789407A			
μPD789417A 子系列	μPD789415A	FD00H 到 FEFFH (512 字节)	FA00H 到 FA1BH (28 字节)	—
	μPD789416A			
	μPD789417A			
	μPD78F9418A			
μPD789800 子系列	μPD789800	FE00H 到 FEFFH (256 字节)	—	—
	μPD78F9801			
μPD789842 子系列	μPD789841	FE00H 到 FEFFH (256 字节)	—	—
	μPD789842			
	μPD78F9842			

1.6 特殊功能寄存器(SFR)区域

片内外部硬件的特殊功能寄存器(SFRs)分配在 FF00H 到 FFFFH 的区域(请参考各产品的用户手册)。

第二章 寄存器

2.1 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。

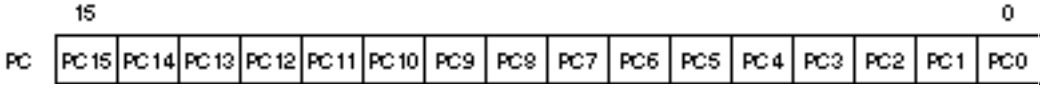
2.1.1 程序计数器(PC)

程序计数器是 16 位的寄存器，可保持程序下次执行的地址信息。

在正常操作中，根据获取指令的字节数 PC 自动增加。当执行跳转指令时，设置立即数和寄存器内容。

当输入$\overline{\text{RESET}}$信号时，程序计数器设置为复位矢量表地址 0000H 和 0001 的值。

图 2-1. 程序计数器的格式



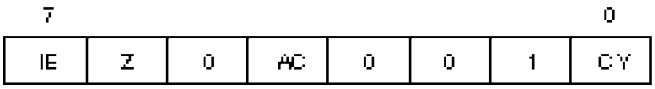
2.1.2 程序状态字(PSW)

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设置或复位。

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 02H。

图 2-2. 程序状态字的格式



(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)，通过优先服务标志(ISP)、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，其他情况置 0。

(3) 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。

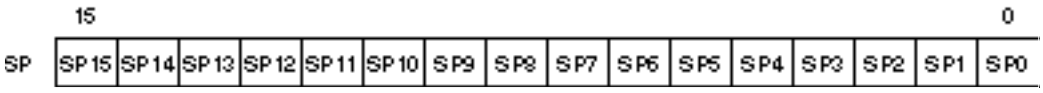
(4) 进位标志(CY)

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。

2.1.3 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。只有内部高速 RAM 区域才能被设置为堆栈区。

图 2-3. 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。

堆栈的数据存储/恢复操作过程如图 2-4 和 2-5 所示。

注意事项　由于复位信号产生时，SP 的内容不确定，所以在使用堆栈前必须先对 SP 初始化。

图 2-4. 将数据存入堆栈

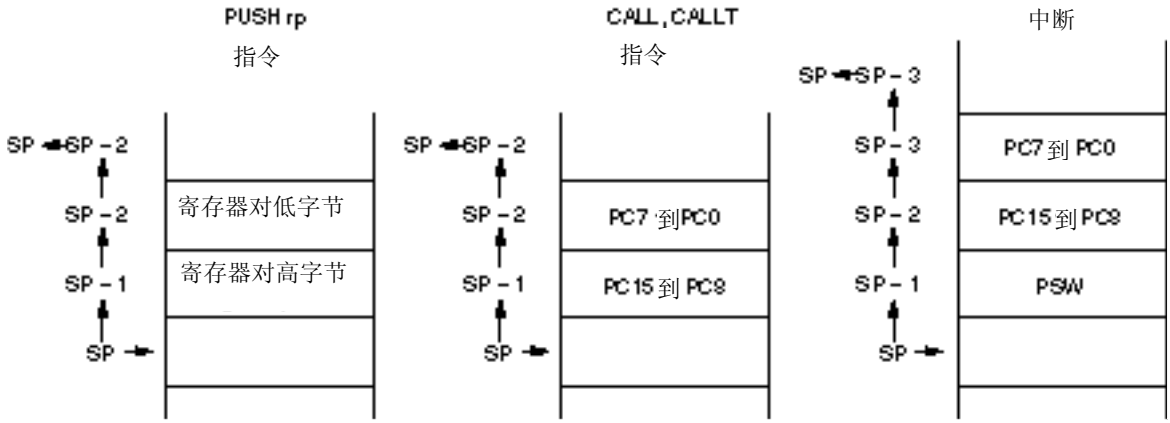
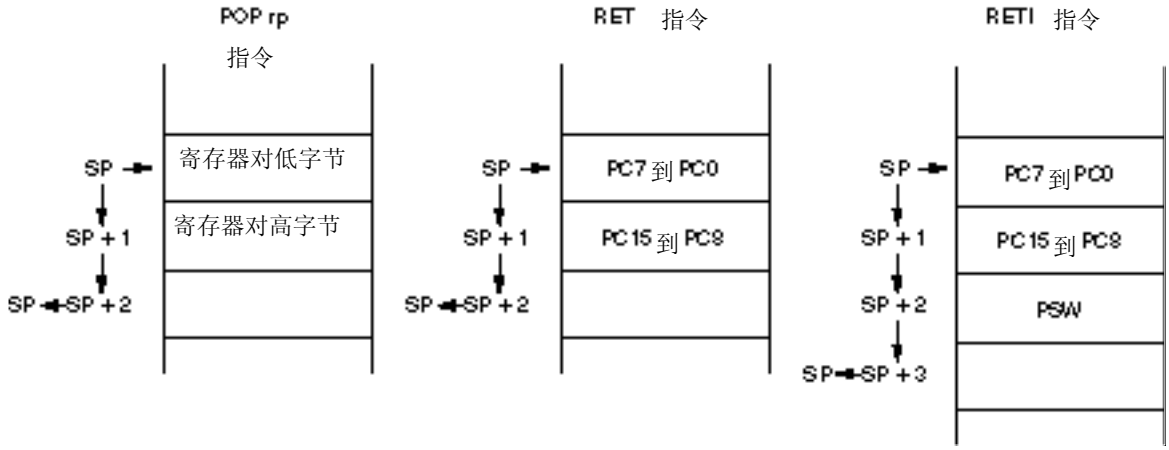


图 2-5. 从堆栈读出数据



2.2 通用寄存器

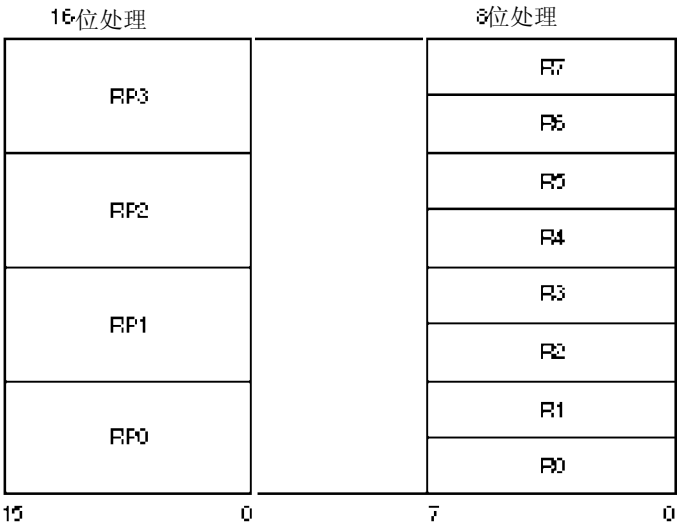
通用寄存器包括 8 个 8 位寄存器(X, A, C, B, E, D, L, 和 H)。

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(AX, BC, DE, 和 HL)使用。

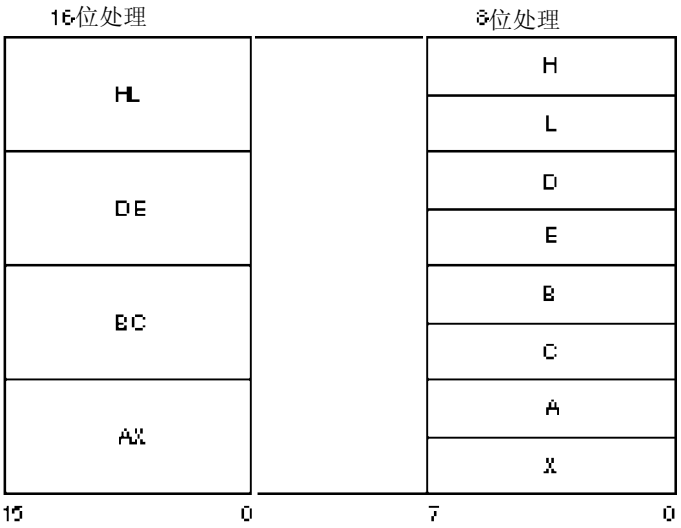
描述通用寄存器时，可以使用功能名称(X, A, C, B, E, D, L, H, AX, BC, DE, 和 HL)或绝对名称(R0 到 R7 和 RP0 到 RP3)。

图 **2-6. 通用寄存器配置**

(a) 绝对名称



(b) 功能名称



2.3 特殊功能寄存器(SFRs)

与通用寄存器不同，特殊功能寄存器具有特定功能，分配在 FF00H 到 FFFFH 的 256 字节的区域。

特殊功能寄存器可像通用寄存器那样用运算指令、传送指令以及位操作指令进行操作。根据特殊功能寄存器的类型不同，可操作的位单元也不同，可以是 1 位、8 位和 16 位。

每种位单元操作的描述如下。

- 1 位操作
  1 位操作指令的操作数(sfr.bit)被描述为汇编程序的保留符号。该操作也可由一个地址来定义。

- 8 位操作
  8 位操作指令的操作数(sfr)被描述为汇编程序的保留符号。该操作也可由一个地址来定义。

- 16 位操作
  16 位操作指令的操作数(sfrp) 被描述为汇编程序的保留符号。寻址时表示为一个偶地址。

关于特殊功能寄存器的详细情况，请参考各产品的用户手册。

[备忘录]

3.1 指令地址寻址

一条指令的地址是由程序计数器(PC)决定的。根据执行指令时所获取的下一条指令字节数，程序计数器(PC)的内容自动增加(每个字节加 1)。在执行转移指令时，将程序计数器(PC)的内容设置为转移目的地址，并按以下寻址方式确定地址。(要了解每条指令的详细信息，请参考 **第五章 指令说明**)。

3.1.1 相对寻址

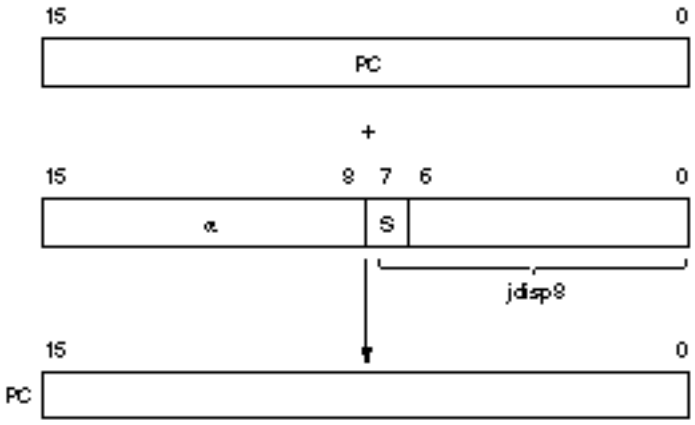
[功能]

将一条指令的 8 位立即数(偏移量: jdisp8)与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。这个偏移量是带符号数的补码(–128 ～ +127)，其中第 7 位是符号位。

换句话说，在相对寻址中，分支的范围是从下一条指令起始地址的-128 到+127 之间。

当执行“BR $addr16”指令或条件转移指令时，将执行相对寻址功能。

[图示]



PC 内容是 BR 指令的下一条指令的起始地址

当S = 0， α 的所有位均为0

当 S = 1， α 的所有位均为 1

3.1.2 立即寻址

[功能]

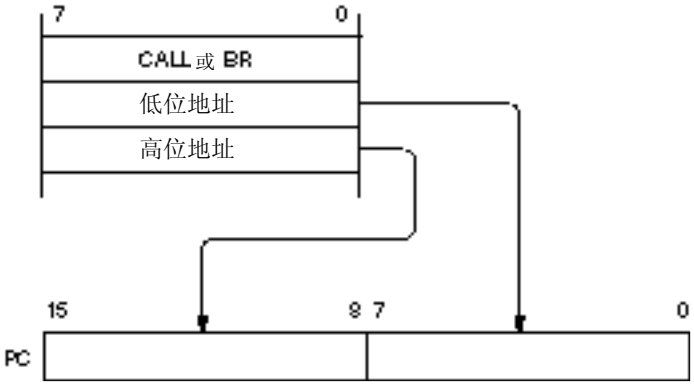
将指令中的立即数赋给程序计数器(PC)，然后转向该地址。

在执行“CALL !addr16”指令或“BR !addr16”指令时，将执行立即寻址功能。

CALL !addr16 和 BR !addr16 指令的转移地址范围是所有存储空间。

[图示]

CALL !addr16 和 BR !addr16 指令



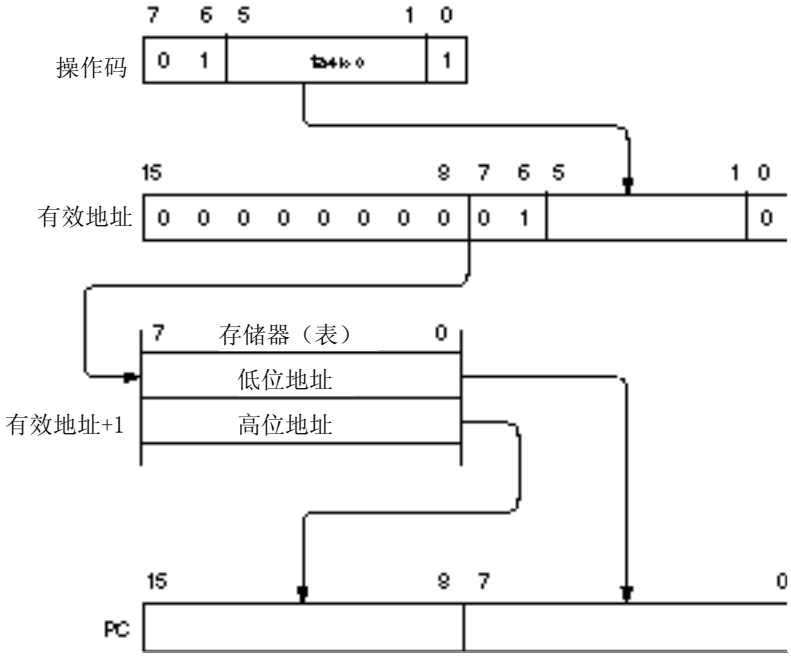
3.1.3 表间接寻址

[功能]

通过指令码低 5 位的立即数(从第 1 位到第 5 位)，访问特定存储区中表的内容(转移目的地址)，并将表的内容赋给程序计数器(PC)，然后转向该地址执行程序。

在执行 CALLT [addr5]指令时，进行表间接寻址。该指令访问的地址范围是表 40H～7FH 中所存储的地址，转移地址范围可以是整个存储器空间。

[图示]



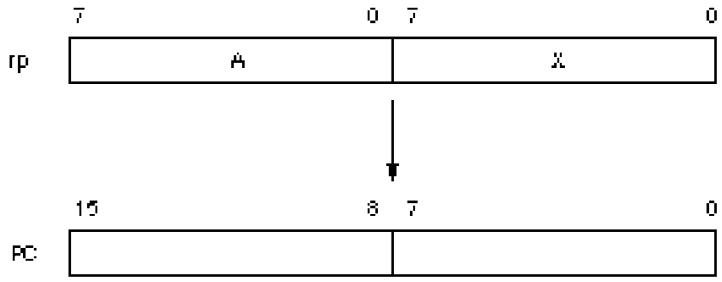
3.1.4 寄存器寻址

[功能]

将寄存器对(AX)的内容赋给程序计数器(PC)，然后转向该地址。

“BR AX”指令将执行寄存器寻址功能。

[图示]



3.2 操作数地址寻址

以下方法用来规定指令执行期间寄存器寻址和存储器寻址所进行的操作。

3.2.1 直接寻址

[功能]

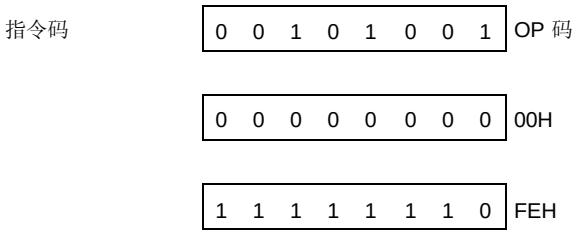
存储器会根据指令字中的操作数地址进行直接寻址操作。

[操作数格式]

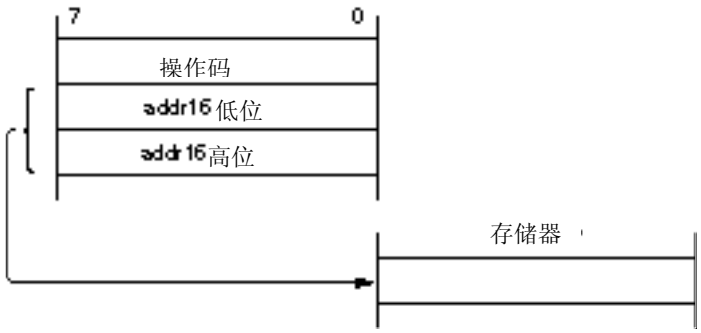
操作数	描述
addr16	标号或 16 位立即数

[描述举例]

MOV A, !FE00H; 将!addr16 设置为 FE00H 时



[图示]



3.2.2 短直接寻址

[功能]

用指令中 8 位立即数直接对存储器的固定操作区域寻址。

该方式的寻址范围是 FE20H～FF1FH 总共 256 字节的区域。内部 RAM 和特殊功能寄存器(SFR)分别映射在 FE20H ～ FEFFH 以及 FF00H ～ FF1FH 的区域。

采用短直接寻址方式的特殊功能寄存器(SFR)区域(FF00H ～ FF1FH)是整个特殊功能寄存器 SFR 区域的一部分。程序中经常访问的端口、用作定时器和事件计数器的比较和捕捉寄存器都被映射到该区域。这些特殊功能寄存器(SFR)可以用很少的字节数和时钟数进行操作。

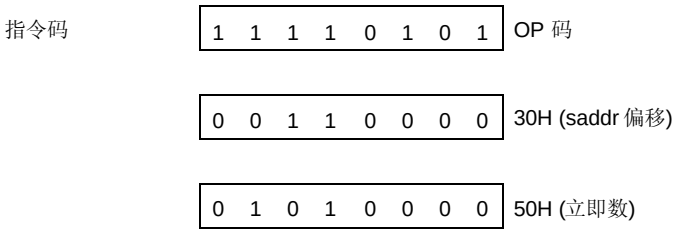
如果 8 位立即数是在 20H 和 FFH 之间，则将一个有效地址的第 8 位设置为 0；如果 8 位立即数是在 00H 与 1FH 之间，则一个有效地址的第 8 位设置为 1。参见下面的**【图示】**。

[操作数格式]

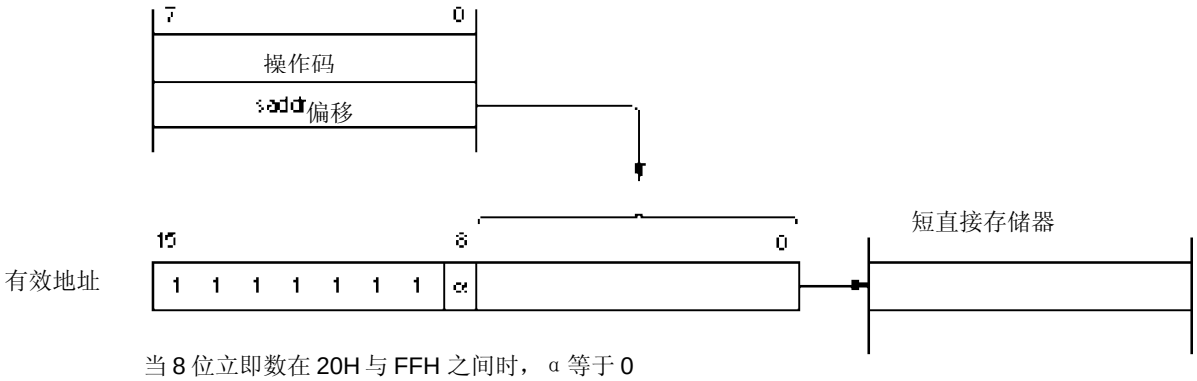
操作数	描述
saddr	标志或从 FE20H ～ FF1FH 的立即数
saddrp	标志或从 FE20H ～ FF1FH 的立即数(仅使用偶地址)

[描述举例]

MOV FE30H, #50H; 当设置 saddr 为 FE30H 和立即数为 50H 时



[图示]



当 8 位立即数在 20H 与 FFH 之间时， α 等于 0

当 8 位立即数的地址在 00H 与 1FH 之间时， α 等于 1

3.2.3 特殊功能寄存器 (SFR) 寻址

[功能]

通过指令中的 8 位立即数对存储器的特殊功能寄存器(SFR)区域进行寻址。

寻址区间为 FF00H～FFCFH 以及 FFE0H～FFFFH，共 240 字节。而映射在 FF00H～FF1FH 区间的特殊功能寄存器则采用短直接寻址方式。

[操作数格式]

操作数	描述
sfr	特殊功能寄存器名称

[描述举例]

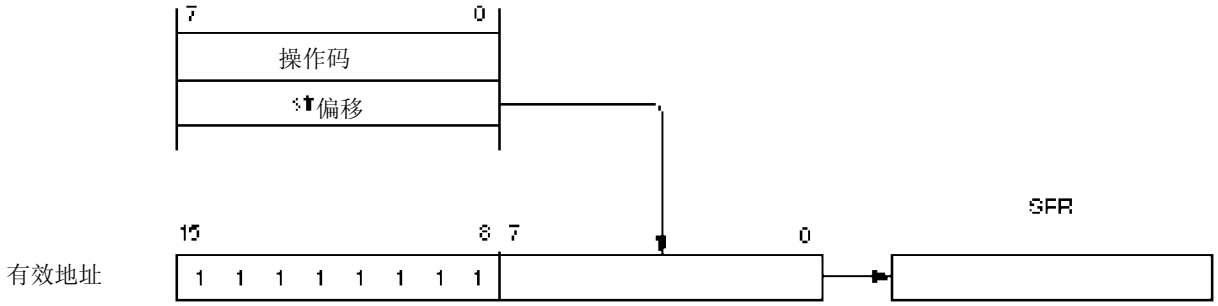
MOV PM0, A; 当选择 PM0 作为 sfr 时

指令码

1	1	1	0	0	1	1	1

0	0	1	0	0	0	0	0

[图示]



3.2.4 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。由指令中的寄存器标识码来指定需要访问的通用寄存器。

当具有下列操作数格式的指令执行时，采用寄存器寻址方式。如果使用 8 位寄存器，则指令码中有 3 位用来表示一个 8 位寄存器。

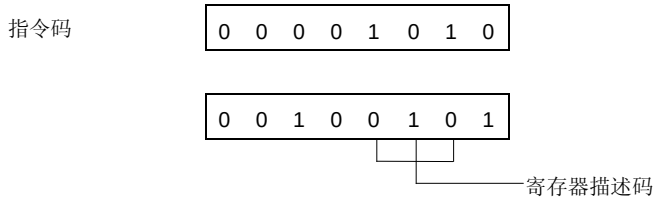
[操作数格式]

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

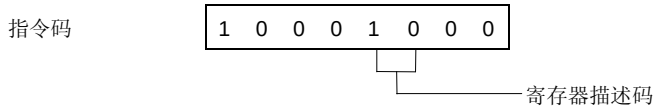
‘r’ 和 ‘rp’可用绝对名称(R0 ～ R7 以及 RP0 ～ RP3)和功能名称(X, A, C, B, E, D, L, H, AX, BC, DE 以及 HL)来描述。

[描述举例]

MOV A, C; 选择 C 寄存器作为 r



INCW DE; 选择 DE 寄存器对作为 rp



3.2.5 寄存器间接寻址

[功能]

根据寄存器对的内容进行寻址。该寄存器对由指令字中的寄存器对指定码指定。这种方式的寻址范围是整个存储空间。

[操作数格式]

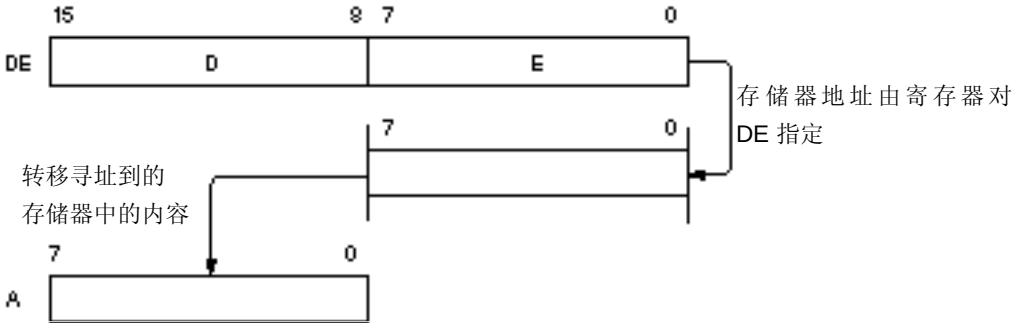
操作数	描述
—	[DE], [HL]

[Description example]

MOV A, [DE]; 选择寄存器对[DE]

指令码	0	0	1	0	1	0	1	1

[图示]



3.2.6 基址寻址

[功能]

将 8 位立即数加到 HL 寄存器对中，HL 寄存器对作为基地址寄存器。根据相加结果寻址。通过将偏移量扩展为 16 位正数，来完成加法操作，第 16 位的进位忽略不计。该寻址方式可对整个存储空间进行。

[操作数格式]

操作数	描述
—	[HL + byte]

[描述举例]

MOV A, [HL+10H]; byte 的值为 10H 时

指令码

0	0	1	0	1	1	0	1

0	0	0	1	0	0	0	0

3.2.7 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。

该方式仅对内部高速 RAM 区域进行寻址。

[描述举例]

以 PUSH DE 指令为例

指令码

1	0	1	0	1	0	1	0

第四章 指令集

本章列出了 78K/0S 系列的指令集。适合所有的 78K/0S 系列产品。

4.1 操作

4.1.1 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[ ]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
- ! ：绝对地址
- $ ：相对地址
- [ ]：间接地址

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

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 4-1. 操作数标识符和标识方法

标识符	标识方法
r rp sfr	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
saddr saddrp	FE20H ～FF1FH 立即数或标号 FE20H ～FF1FH 立即数或标号(仅用于偶地址)
addr16 addr5	0000H ～FFFFH 立即数或标号(仅用于 16 位数据传送指令的偶地址) 0040H ～007FH 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

备注 特殊功能寄存器符号请参考各产品的用户手册。

4.1.2 操作栏描述

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
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
IE: 中断请求允许标志
NMIS: 不可屏蔽中断服务标志
( ): 括号中的地址或寄存器所指的存储单元的内容
X_H, X_L: 16 位寄存器的高 8 位和低 8 位
∧: 逻辑与(AND)
∨: 逻辑或(OR)
∀: 逻辑异或(exclusive OR)
$\overline{\quad}$: 数据取反
addr16: 16 位立即数或标号
jdisp8: 带符号的 8 位数据（偏移量）

4.1.3 标志操作栏的描述

(空): 不受影响
0: 清零
1: 设置为 1
×: 根据结果进行设置/清零
R: 恢复先前保存的值

4.1.4 时钟栏描述

指令执行期间时钟周期数如下所示。
一个指令时钟周期等于由处理器时钟控制寄存器(PCC)所选择的一个 CPU 时钟周期(f_{CPU})。

操作列表如下所示。

4.1.5 操作列表

助记符	操作数	字节	时钟	操作	标志		
					Z	AC	CY
MOV	r, #byte	3	6	r ← byte			
	saddr, #byte	3	6	(saddr) ← byte			
	sfr, #byte	3	6	sfr ← byte			
	A, r 注1	2	4	A ← r			
	r, A 注1	2	4	r ← A			
	A, saddr	2	4	A ← (saddr)			
	saddr, A	2	4	(saddr) ← A			
	A, sfr	2	4	A ← sfr			
	sfr, A	2	4	sfr ← A			
	A, !addr16	3	8	A ← (addr16)			
	!addr16, A	3	8	(addr16) ← A			
	PSW, #byte	3	6	PSW ← byte	×	×	×
	A, PSW	2	4	A ← PSW			
	PSW, A	2	4	PSW ← A	×	×	×
	A, [DE]	1	6	A ← (DE)			
	[DE], A	1	6	(DE) ← A			
	A, [HL]	1	6	A ← (HL)			
	[HL], A	1	6	(HL) ← A			
	A, [HL + byte]	2	6	A ← (HL + byte)			
	[HL + byte], A	2	6	(HL + byte) ← A			
XCH	A, X	1	4	A ↔ X			
	A, r 注2	2	6	A ↔ r			
	A, saddr	2	6	A ↔ (saddr)			
	A, sfr	2	6	A ↔ sfr			
	A, [DE]	1	8	A ↔ (DE)			
	A, [HL]	1	8	A ↔ (HL)			
	A, [HL + byte]	2	8	A ↔ (HL + byte)			

注 **1.** r = A 除外

2. r = A, X 除外

备注 一个指令时钟周期是指由处理器时钟控制寄存器（PCC）选择的 CPU 时钟（f_{CPU}）的一个周期。

助记符	操作数	字节	时钟	操作	标志		
					Z	AC	CY
MOVW	rp, #word	3	6	rp ← word			
	AX, saddrp	2	6	AX ← (saddrp)			
	saddrp, AX	2	8	(saddrp) ← AX			
	AX, rp 注	1	4	AX ← rp			
	rp, AX 注	1	4	rp ← AX			
XCHW	AX, rp 注	1	8	AX ↔ rp			
ADD	A, #byte	2	4	A, CY ← A + byte	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) + byte	×	×	×
	A, r	2	4	A, CY ← A + r	×	×	×
	A, saddr	2	4	A, CY ← A + (saddr)	×	×	×
	A, !addr16	3	8	A, CY ← A + (addr16)	×	×	×
	A, [HL]	1	6	A, CY ← A + (HL)	×	×	×
	A, [HL + byte]	2	6	A, CY ← A + (HL + byte)	×	×	×
ADDC	A, #byte	2	4	A, CY ← A + byte + CY	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) + byte + CY	×	×	×
	A, r	2	4	A, CY ← A + r + CY	×	×	×
	A, saddr	2	4	A, CY ← A + (saddr) + CY	×	×	×
	A, !addr16	3	8	A, CY ← A + (addr16) + CY	×	×	×
	A, [HL]	1	6	A, CY ← A + (HL) + CY	×	×	×
	A, [HL + byte]	2	6	A, CY ← A + (HL + byte) + CY	×	×	×
SUB	A, #byte	2	4	A, CY ← A – byte	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) – byte	×	×	×
	A, r	2	4	A, CY ← A – r	×	×	×
	A, saddr	2	4	A, CY ← A – (saddr)	×	×	×
	A, !addr16	3	8	A, CY ← A – (addr16)	×	×	×
	A, [HL]	1	6	A, CY ← A – (HL)	×	×	×
	A, [HL + byte]	2	6	A, CY ← A – (HL + byte)	×	×	×

注 仅当 rp = BC, DE, 或 HL

备注 一个指令时钟周期是指由处理器时钟控制寄存器（PCC）选择的 CPU 时钟（f_{CPU}）的一个周期。

助记符	操作数	字节	时钟	操作	标志		
					Z	AC	CY
SUBC	A, #byte	2	4	A, CY ← A – byte – CY	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) – byte – CY	×	×	×
	A, r	2	4	A, CY ← A – r – CY	×	×	×
	A, saddr	2	4	A, CY ← A – (saddr) – CY	×	×	×
	A, !addr16	3	8	A, CY ← A – (addr16) – CY	×	×	×
	A, [HL]	1	6	A, CY ← A – (HL) – CY	×	×	×
	A, [HL + byte]	2	6	A, CY ← A – (HL + byte) – CY	×	×	×
AND	A, #byte	2	4	A ← A∧byte	×		
	saddr, #byte	3	6	(saddr) ← (saddr)∧byte	×		
	A, r	2	4	A ← A∧r	×		
	A, saddr	2	4	A ← A∧(saddr)	×		
	A, !addr16	3	8	A ← A∧(addr16)	×		
	A, [HL]	1	6	A ← A∧(HL)	×		
	A, [HL + byte]	2	6	A ← A∧(HL + byte)	×		
OR	A, #byte	2	4	A ← A∨byte	×		
	saddr, #byte	3	6	(saddr) ← (saddr)∨byte	×		
	A, r	2	4	A ← A∨r	×		
	A, saddr	2	4	A ← A∨(saddr)	×		
	A, !addr16	3	8	A ← A∨(addr16)	×		
	A, [HL]	1	6	A ← A∨(HL)	×		
	A, [HL + byte]	2	6	A ← A∨(HL + byte)	×		
XOR	A, #byte	2	4	A ← A⩛byte	×		
	saddr, #byte	3	6	(saddr) ← (saddr)⩛byte	×		
	A, r	2	4	A ← A⩛r	×		
	A, saddr	2	4	A ← A⩛(saddr)	×		
	A, !addr16	3	8	A ← A⩛(addr16)	×		
	A, [HL]	1	6	A ← A⩛(HL)	×		
	A, [HL + byte]	2	6	A ← A⩛(HL + byte)	×		
CMP	A, #byte	2	4	A – byte	×	×	×
	saddr, #byte	3	6	(saddr) – byte	×	×	×
	A, r	2	4	A – r	×	×	×
	A, saddr	2	4	A – (saddr)	×	×	×
	A, !addr16	3	8	A – (addr16)	×	×	×
	A, [HL]	1	6	A – (HL)	×	×	×
	A, [HL + byte]	2	6	A – (HL + byte)	×	×	×

备注 一个指令时钟周期是指由处理器时钟控制寄存器（PCC）选择的 CPU 时钟（f_{CPU}）的一个周期。

助记符	操作数	字节	时钟	操作	标志		
					Z	AC	CY
ADDW	AX, #word	3	6	AX, CY ← AX + word	×	×	×
SUBW	AX, #word	3	6	AX, CY ← AX – word	×	×	×
CMPW	AX, #word	3	6	AX – word	×	×	×
INC	r	2	4	r ← r + 1	×	×	
	saddr	2	4	(saddr) ← (saddr) + 1	×	×	
DEC	r	2	4	r ← r – 1	×	×	
	saddr	2	4	(saddr) ← (saddr) – 1	×	×	
INCW	rp	1	4	rp ← rp + 1			
DECW	rp	1	4	rp ← rp – 1			
ROR	A, 1	1	2	(CY, A_7 ← A_0, A_{m-1} ← A_m) × 1			×
ROL	A, 1	1	2	(CY, A_0 ← A_7, A_{m+1} ← A_m) × 1			×
RORC	A, 1	1	2	(CY ← A_0, A_7 ← CY, A_{m-1} ← A_m) × 1			×
ROLC	A, 1	1	2	(CY ← A_7, A_0 ← CY, A_{m+1} ← A_m) × 1			×
SET1	saddr.bit	3	6	(saddr.bit) ← 1			
	sfr.bit	3	6	sfr.bit ← 1			
	A.bit	2	4	A.bit ← 1			
	PSW.bit	3	6	PSW.bit ← 1	×	×	×
	[HL].bit	2	10	(HL).bit ← 1			
CLR1	saddr.bit	3	6	(saddr.bit) ← 0			
	sfr.bit	3	6	sfr.bit ← 0			
	A.bit	2	4	A.bit ← 0			
	PSW.bit	3	6	PSW.bit ← 0	×	×	×
	[HL].bit	2	10	(HL).bit ← 0			
SET1	CY	1	2	CY ← 1			1
CLR1	CY	1	2	CY ← 0			0
NOT1	CY	1	2	CY ← $\overline{CY}$			×
CALL	!addr16	3	6	(SP – 1) ← (PC + 3)$_H$, (SP – 2) ← (PC + 3)$_L$, PC ← addr16, SP ← SP – 2			
CALLT	[addr5]	1	8	(SP – 1) ← (PC + 1)$_H$, (SP – 2) ← (PC + 1)$_L$, PC_H ← (00000000, addr5 + 1), PC_L ← (00000000, addr5), SP ← SP – 2			

备注 一个指令时钟周期是指由处理器时钟控制寄存器（PCC）选择的 CPU 时钟（f_{CPU}）的一个周期。

助记符	操作数	字节	时钟	操作	标志		
					Z	AC	CY
RET		1	6	$PC_H \leftarrow (SP + 1), PC_L \leftarrow (SP), SP \leftarrow SP + 2$			
RETI		1	8	$PC_H \leftarrow (SP + 1), PC_L \leftarrow (SP)$, $PSW \leftarrow (SP + 2), SP \leftarrow SP + 3, NMIS \leftarrow 0$	R	R	R
PUSH	PSW	1	2	$(SP - 1) \leftarrow PSW, SP \leftarrow SP - 1$			
	rp	1	4	$(SP - 1) \leftarrow rp_H, (SP - 2) \leftarrow rp_L, SP \leftarrow SP - 2$			
POP	PSW	1	4	$PSW \leftarrow (SP), SP \leftarrow SP + 1$	R	R	R
	rp	1	6	$rp_H \leftarrow (SP + 1), rp_L \leftarrow (SP), SP \leftarrow SP + 2$			
MOVW	SP,AX	2	8	$SP \leftarrow AX$			
	AX,SP	2	6	$AX \leftarrow SP$			
BR	!addr16	3	6	$PC \leftarrow$ addr16			
	$addr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8			
	AX	1	6	$PC_H \leftarrow A, PC_L \leftarrow X$			
BC	$addr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 如果 CY = 1			
BNC	$addr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 如果 CY = 0			
BZ	$addr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 如果 Z = 1			
BNZ	$addr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 如果 Z = 0			
BT	saddr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 如果(saddr.bit) = 1			
	sfr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 如果 sfr.bit = 1			
	A.bit, $addr16	3	8	$PC \leftarrow PC + 3 +$ jdisp8 如果 A.bit = 1			
	PSW.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 如果 PSW.bit = 1			
BF	saddr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 如果(saddr.bit) = 0			
	sfr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 如果 sfr.bit = 0			
	A.bit, $addr16	3	8	$PC \leftarrow PC + 3 +$ jdisp8 如果 A.bit = 0			
	PSW.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 如果 PSW.bit = 0			
DBNZ	B, $addr16	2	6	$B \leftarrow B - 1$, $PC \leftarrow PC + 2 +$ jdisp8 如果 $B \neq 0$			
	C, $addr16	2	6	$C \leftarrow C - 1$, $PC \leftarrow PC + 2 +$ jdisp8 如果 $C \neq 0$			
	saddr, $addr16	3	8	(saddr) $\leftarrow$ (saddr) – 1, $PC \leftarrow PC + 3 +$ jdisp8 如果(saddr) $\neq 0$			
NOP		1	2	无操作			
EI		3	6	IE $\leftarrow$ 1 (允许中断)			
DI		3	6	IE $\leftarrow$ 0 (禁止中断)			
HALT		1	2	设置 HALT 模式			
STOP		1	2	设置 STOP 模式			

备注 一个指令时钟周期是指由处理器时钟控制寄存器（PCC）选择的 CPU 时钟（f_{CPU}）的一个周期。

4.1.6 按寻址类型列出指令

(1) 8 位指令

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, INC, DEC, ROR, ROL, RORC, ROLC, PUSH, POP, DBNZ

第一操作数 \ 第二操作数	#byte	A	r	sfr	saddr	!addr16	PSW	[DE]	[HL]	[HL + byte]	$addr16	1	无
A	ADD ADDC SUB SUBC AND OR XOR CMP		MOV 注 XCH 注 ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP		ROR ROL RORC ROLC	
r	MOV	MOV 注											INC DEC
B, C											DBNZ		
sfr	MOV	MOV											
saddr	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV									DBNZ		INC DEC
!addr16		MOV											
PSW	MOV	MOV											PUSH POP
[DE]		MOV											
[HL]		MOV											
[HL + byte]		MOV											

注 r = A 除外

(2) 16 位指令

MOVW, XCHW, ADDW, SUBW, CMPW, PUSH, POP, INCW, DECW

第二操作数 / 第一操作数	#word	AX	rp 注	saddrp	SP	无
AX	ADDW SUBW CMPW		MOVW XCHW	MOVW	MOVW	
rp	MOVW	MOVW 注				INCW DECW PUSH POP
saddrp		MOVW				
SP		MOVW				

注 仅当 rp = BC, DE, HL

(3) 位操作指令

SET1, CLR1, NOT1, BT, BF

第二操作数 / 第一操作数	$saddr	无
A.bit	BT BF	SET1 CLR1
sfr.bit	BT BF	SET1 CLR1
saddr.bit	BT BF	SET1 CLR1
PSW.bit	BT BF	SET1 CLR1
[HL].bit		SET1 CLR1
CY		SET1 CLR1 NOT1

(4) 调用指令 / 转移指令

CALL, CALLT, BR, BC, BNC, BZ, BNZ, BT, BF, DBNZ

第 1 操作数 \ 第 2 操作数	AX	!addr16	[addr5]	$addr16
基本指令	BR	CALL BR	CALLT	BR BC BNC BZ BNZ
复合指令				DBNZ

(5) 其它指令

RET, RETI, NOP, EI, DI, HALT, STOP

4.2 指令码

4.2.1 指令码表描述

r

R_2	R_1	R_0	寄存器	
0	0	0	R0	X
0	0	1	R1	A
0	1	0	R2	C
0	1	1	R3	B
1	0	0	R4	E
1	0	1	R5	D
1	1	0	R6	L
1	1	1	R7	H

rp

P_1	P_0	寄存器对	
0	0	RP0	AX
0	1	RP1	BC
1	0	RP2	DE
1	1	RP3	HL

Bn: 立即数“位”
数据: 8位立即数“字节”
低/高字节: 16 位立即数“字”
Saddr 偏移: 16 位地址低 8 位偏移数“saddr”
Sfr 偏移: sfr16 位地址低 8 位偏移数
低/高地址: 16 位立即数“addr16”
jdisp: 带符号的 8 位数据，开始地址与跳转到下一条指令地址间的相对地址
$ta_{4\ to\ 0}$: 5位立即数“addr5”

4.2.2 指令码列表

助记符	操作数	指令码			
		B1	B2	B3	B4
MOV	r, #byte	0 0 0 0 1 0 1 0	1 1 1 1 $R_2R_1R_0$1	数据	
	saddr, #byte	1 1 1 1 0 1 0 1	Saddr 偏移	数据	
	sfr, #byte	1 1 1 1 0 1 1 1	Sfr 偏移	数据	
	A, r 注1	0 0 0 0 1 0 1 0	0 0 1 0 $R_2R_1R_0$1		
	r, A 注1	0 0 0 0 1 0 1 0	1 1 1 0 $R_2R_1R_0$1		
	A, saddr	0 0 1 0 0 1 0 1	Saddr 偏移		
	saddr, A	1 1 1 0 0 1 0 1	Saddr 偏移		
	A, sfr	0 0 1 0 0 1 1 1	Sfr 偏移		
	sfr, A	1 1 1 0 0 1 1 1	Sfr 偏移		
	A, !addr16	0 0 1 0 1 0 0 1	低地址	高地址	
	!addr16, A	1 1 1 0 1 0 0 1	低地址	高地址	
★	PSW, #byte	1 1 1 1 0 1 0 1	0 0 0 1 1 1 1 0	数据	
	A, PSW	0 0 1 0 0 1 0 1	0 0 0 1 1 1 1 0		
	PSW, A	1 1 1 0 0 1 0 1	0 0 0 1 1 1 1 0		
	A, [DE]	0 0 1 0 1 0 1 1			
	[DE], A	1 1 1 0 1 0 1 1			
	A, [HL]	0 0 1 0 1 1 1 1			
	[HL], A	1 1 1 0 1 1 1 1			
	A, [HL + byte]	0 0 1 0 1 1 0 1	数据		
	[HL + byte], A	1 1 1 0 1 1 0 1	数据		
XCH	A, X	1 1 0 0 0 0 0 0			
	A, r 注2	0 0 0 0 1 0 1 0	0 0 0 0 $R_2R_1R_0$1		
	A, saddr	0 0 0 0 0 1 0 1	Saddr 偏移		
	A, sfr	0 0 0 0 0 1 1 1	Sfr 偏移		
	A, [DE]	0 0 0 0 1 0 1 1			
	A, [HL]	0 0 0 0 1 1 1 1			
	A, [HL + byte]	0 0 0 0 1 1 0 1	数据		
MOVW	rp, #word	1 1 1 1 $P_1P_0$0 0	低字节	高字节	
	AX, saddrp	1 1 0 1 0 1 1 0	Saddr 偏移		
	saddrp, AX	1 1 1 0 0 1 1 0	Saddr 偏移		
	AX, rp 注3	1 1 0 1 $P_1P_0$0 0			
★	rp, AX 注3	1 1 1 0 $P_1P_0$0 0			
XCHW	AX, rp 注3	1 1 0 0 $P_1P_0$0 0			

注 **1.** r = A 除外

2. r = A, X 除外

3. 仅当 rp = BC, DE, 或 HL

助记符	操作数	指令码			
		B1	B2	B3	B4
ADD	A, #byte	1 0 0 0 0 0 1 1	数据		
	saddr, #byte	1 0 0 0 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	1 0 0 0 $R_2R_1R_0$1		
	A, saddr	1 0 0 0 0 1 0 1	Saddr 偏移		
	A, !addr16	1 0 0 0 1 0 0 1	低地址	高地址	
	A, [HL]	1 0 0 0 1 1 1 1			
	A, [HL + byte]	1 0 0 0 1 1 0 1	数据		
ADDC	A, #byte	1 0 1 0 0 0 1 1	数据		
	saddr, #byte	1 0 1 0 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	1 0 1 0 $R_2R_1R_0$1		
	A, saddr	1 0 1 0 0 1 0 1	Saddr 偏移		
	A, !addr16	1 0 1 0 1 0 0 1	低地址	高地址	
	A, [HL]	1 0 1 0 1 1 1 1			
	A, [HL + byte]	1 0 1 0 1 1 0 1	数据		
SUB	A, #byte	1 0 0 1 0 0 1 1	数据		
	saddr, #byte	1 0 0 1 0 0 0 1	Saddr-offset	数据	
	A, r	0 0 0 0 1 0 1 0	1 0 0 1 $R_2R_1R_0$1		
	A, saddr	1 0 0 1 0 1 0 1	Saddr 偏移		
	A, !addr16	1 0 0 1 1 0 0 1	低地址	高地址	
	A, [HL]	1 0 0 1 1 1 1 1			
	A, [HL + byte]	1 0 0 1 1 1 0 1	数据		
SUBC	A, #byte	1 0 1 1 0 0 1 1	数据		
	saddr, #byte	1 0 1 1 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	1 0 1 1 $R_2R_1R_0$1		
	A, saddr	1 0 1 1 0 1 0 1	Saddr 偏移		
	A, !addr16	1 0 1 1 1 0 0 1	低地址	高地址	
	A, [HL]	1 0 1 1 1 1 1 1			
	A, [HL + byte]	1 0 1 1 1 1 0 1	数据		
AND	A, #byte	0 1 1 0 0 0 1 1	数据		
	saddr, #byte	0 1 1 0 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	0 1 1 0 $R_2R_1R_0$1		
	A, saddr	0 1 1 0 0 1 0 1	Saddr 偏移		
	A, !addr16	0 1 1 0 1 0 0 1	低地址	高地址	
	A, [HL]	0 1 1 0 1 1 1 1			
	A, [HL + byte]	0 1 1 0 1 1 0 1	数据		

助记符	操作数	指令码			
		B1	B2	B3	B4
OR	A, #byte	0 1 1 1 0 0 1 1	数据		
	saddr, #byte	0 1 1 1 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	0 1 1 1 $R_2R_1R_0$1		
	A, saddr	0 1 1 1 0 1 0 1	Saddr 偏移		
	A, !addr16	0 1 1 1 1 0 0 1	低地址	高地址	
	A, [HL]	0 1 1 1 1 1 1 1			
	A, [HL + byte]	0 1 1 1 1 1 0 1	数据		
XOR	A, #byte	0 1 0 0 0 0 1 1	数据		
	saddr, #byte	0 1 0 0 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	0 1 0 0 $R_2R_1R_0$1		
	A, saddr	0 1 0 0 0 1 0 1	Saddr 偏移		
	A, !addr16	0 1 0 0 1 0 0 1	低地址	高地址	
	A, [HL]	0 1 0 0 1 1 1 1			
	A, [HL + byte]	0 1 0 0 1 1 0 1	数据		
CMP	A, #byte	0 0 0 1 0 0 1 1	数据		
	saddr, #byte	0 0 0 1 0 0 0 1	Saddr 偏移	数据	
	A, r	0 0 0 0 1 0 1 0	0 0 0 1 $R_2R_1R_0$1		
	A, saddr	0 0 0 1 0 1 0 1	Saddr 偏移		
	A, !addr16	0 0 0 1 1 0 0 1	低地址	高地址	
	A, [HL]	0 0 0 1 1 1 1 1			
	A, [HL + byte]	0 0 0 1 1 1 0 1	数据		
ADDW	AX, #word	1 1 0 1 0 0 1 0	低字节	高字节	
SUBW	AX, #word	1 1 0 0 0 0 1 0	低字节	高字节	
CMPW	AX, #word	1 1 1 0 0 0 1 0	低字节	高字节	
INC	r	0 0 0 0 1 0 1 0	1 1 0 0 $R_2R_1R_0$1		
	saddr	1 1 0 0 0 1 0 1	Saddr 偏移		
DEC	r	0 0 0 0 1 0 1 0	1 1 0 1 $R_2R_1R_0$1		
	saddr	1 1 0 1 0 1 0 1	Saddr 偏移		
INCW	rp	1 0 0 0 $P_1P_0$0 0			
DECW	rp	1 0 0 1 $P_1P_0$0 0			
ROR	A, 1	0 0 0 0 0 0 0 0			
ROL	A, 1	0 0 0 1 0 0 0 0			
RORC	A, 1	0 0 0 0 0 0 1 0			
ROLC	A, 1	0 0 0 1 0 0 1 0			

助记符	操作数	指令码			
		B1	B2	B3	B4
SET1	saddr.bit	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 1 0 1 0	Saddr 偏移	
	sfr.bit	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 0 1 1 0	Sfr 偏移	
	A.bit	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 0 0 1 0		
	PSW.bit	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 1 0 1 0	0 0 0 1 1 1 1 0	
	[HL].bit	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 1 1 1 0		
CLR1	saddr.bit	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 1 0 1 0	Saddr 偏移	
	sfr.bit	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 0 1 1 0	Sfr 偏移	
	A.bit	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 0 0 1 0		
	PSW.bit	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 1 0 1 0	0 0 0 1 1 1 1 0	
	[HL].bit	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 1 1 1 0		
SET1	CY	0 0 0 1 0 1 0 0			
CLR1	CY	0 0 0 0 0 1 0 0			
NOT1	CY	0 0 0 0 0 1 1 0			
CALL	!addr16	0 0 1 0 0 0 1 0	低地址	高地址	
CALLT	[addr5]	0 1 $ta_{4\ to\ 0}$ 0			
RET		0 0 1 0 0 0 0 0			
RETI		0 0 1 0 0 1 0 0			
PUSH	PSW	0 0 1 0 1 1 1 0			
	rp	1 0 1 0 P_1 P_0 1 0			
POP	PSW	0 0 1 0 1 1 0 0			
	rp	1 0 1 0 P_1 P_0 0 0			
MOVW	SP, AX	1 1 1 0 0 1 1 0	0 0 0 1 1 1 0 0		
	AX, SP	1 1 0 1 0 1 1 0	0 0 0 1 1 1 0 0		
BR	!addr16	1 0 1 1 0 0 1 0	低地址	高地址	
	$addr16	0 0 1 1 0 0 0 0	jdisp		
	AX	1 0 1 1 0 0 0 0			
BC	$addr16	0 0 1 1 1 0 0 0	jdisp		
BNC	$addr16	0 0 1 1 1 0 1 0	jdisp		
BZ	$addr16	0 0 1 1 1 1 0 0	jdisp		
BNZ	$addr16	0 0 1 1 1 1 1 0	jdisp		
BT	saddr.bit, $addr16	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 1 0 0 0	Saddr 偏移	jdisp
	sfr.bit, $addr16	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 0 1 0 0	Sfr 偏移	jdisp
	A.bit, $addr16	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 0 0 0 0	jdisp	
	PSW.bit, $addr16	0 0 0 0 1 0 1 0	1 B_2 B_1 B_0 1 0 0 0	0 0 0 1 1 1 1 0	jdisp

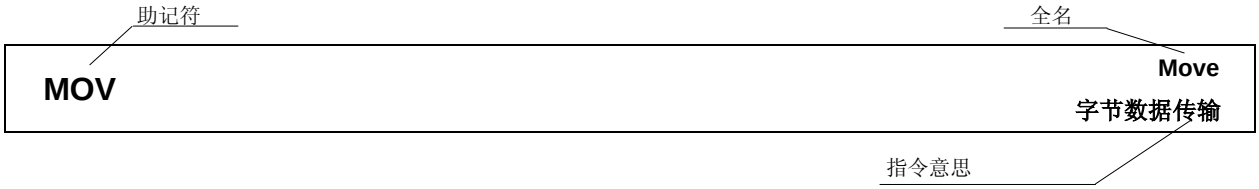
助记符	操作数	指令码			
		B1	B2	B3	B4
BF	saddr.bit, $addr16	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 1 0 0 0	Saddr 偏移	jdisp
	sfr.bit, $addr16	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 0 1 0 0	Sfr 偏移	jdisp
	A.bit, $addr16	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 0 0 0 0	jdisp	
	PSW.bit, $addr16	0 0 0 0 1 0 1 0	0 B_2 B_1 B_0 1 0 0 0	0 0 0 1 1 1 1 0	jdisp
DBNZ	B, $addr16	0 0 1 1 0 1 1 0	jdisp		
	C, $addr16	0 0 1 1 0 1 0 0	jdisp		
	saddr, $addr16	0 0 1 1 0 0 1 0	Saddr 偏移	jdisp	
NOP		0 0 0 0 1 0 0 0			
EI		0 0 0 0 1 0 1 0	0 1 1 1 1 0 1 0	0 0 0 1 1 1 1 0	
DI		0 0 0 0 1 0 1 0	1 1 1 1 1 0 1 0	0 0 0 1 1 1 1 0	
HALT		0 0 0 0 1 1 0 0			
STOP		0 0 0 0 1 1 1 0			

第五章 指令描述

本章解释了 78K/0S 系列的指令。每一条指令用助记符表示，包括若干操作数的描述。
指令的基本配置描述在下一页。
如需了解指令字节数和操作码，请参考 **第四章 指令集。**

所有指令都适用于 78K/0S 系列产品。

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 **4.1 操作** 中每个操作数符号的的描述。

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

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] 详细地描述指令操作。

- 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。

[描述例子]

MOV A, #4DH; 4DH 传送到 A 寄存器。

5.1 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
	A, r 注
	r, A 注
	A, saddr
	saddr, A
	A, sfr
	sfr, A
	A, !addr16

助记符	操作数(dst, src)
MOV	!addr16, A
	PSW, #byte
	A, PSW
	PSW, A
	A, [DE]
	[DE], A
	A, [HL]
	[HL], A
	A, [HL + byte]
	[HL + byte], A

注 r = A 除外

[标志]

PSW, #byte 和 PSW, A 操作数

Z	AC	CY
×	×	×

所有其他操作数

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。
- 没有中断在“MOV PSW, #byte” 指令或者“MOV PSW, A” 指令和如下的指令中确认。

[描述示例]

MOV A, #4DH; 4DH 传输到 A 寄存器。

XCH

交换

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, X
	A, r 注
	A, saddr
	A, sfr
	A, [DE]
	A, [HL]
	A, [HL + byte]

注 r = A, X 除外

[标志]

Z	AC	CY

[描述]

- 第一个和第二个操作数的内容交换。

[描述示例]

XCH A, 0FEBCH; A 寄存器内容和地址 FEBCH 中的内容交换。

5.2 16 位数据传输指令

如下指令是 16 位数据传输指令。

MOVW

移动字

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
	AX, saddrp
	saddrp, AX
	AX, rp 注
	rp, AX 注

注 仅当 rp = BC, DE 或 HL

[符号]

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 寄存器内容传输到 AX 寄存器。

[注意事项]

只有偶地址指定到 saddrp，一个奇地址不能被指定。

XCHW

交换字
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。

5.3 8 位操作指令

如下是 8 位操作指令。

ADD

加

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
	A, saddr

助记符	操作数(dst, src)
ADD	A, !addr16
	A, [HL]
	A, [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 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)。其他情况下，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)。其他情况下，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)。其他情况下，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

带进位的加

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
	A, saddr

助记符	操作数(dst, src)
ADDC	A, !addr16
	A, [HL]
	A, [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)的内容并且加上 CY，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)。其他情况下，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)。其他情况下，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)。其他情况下，AC 标志清零(0)。

[描述示例]

ADDC A, [HL]; A 寄存器内容，地址（HL 寄存器）中的内容，和 CY 标志相加，结果存储在 A 寄存器中。

SUB

减

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
	A, saddr

助记符	操作数(dst, src)
SUB	A, !addr16
	A, [HL]
	A, [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)的内容，结果存储在 CY 标志和目的操作数(dst) 中。
- 如果减的结果显示 dst 是 0，Z 标志设置为(1)。其他情况下，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)。其他情况下，CY 标志清零(0)。
- 如果减法从第 4 位到第 3 位产生一个借位，AC 标志被设置为 (1)。其他情况下，AC 标志清零(0)。

[描述示例]

SUB A, D; D 寄存器从 A 寄存器中减去，并且结果存储在 A 寄存器中。

SUBC

带借位减

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
	A, saddr

助记符	操作数(dst, src)
SUBC	A, !addr16
	A, [HL]
	A, [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 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。
- 如果减的结果显示 dst 是 0，Z 标志设置为(1)。其他情况下，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)。其他情况下，CY 标志清零(0)。
- 如果减法从第 4 位到第 3 位产生一个借位，AC 标志被设置为 (1)。其他情况下，AC 标志清零(0)。

[描述示例]

SUBC A, [HL]; 地址 (HL 寄存器)的内容和 CY 标志从 A 寄存器中减去，并且结果存储在 A 寄存器中。

AND

与

字节数据的逻辑乘积

[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
	A, saddr

助记符	操作数(dst, src)
AND	A, !addr16
	A, [HL]
	A, [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)。其他情况下，Z 标志清零。

[描述示例]

AND 0FEBAH, #11011100B; 地址 FEBAH 中的内容和 11011100B 以位相与，并且结果存储在地址 FEBAH 中。

OR

或

字节数据的逻辑和

[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
	A, saddr

助记符	操作数(dst, src)
OR	A, !addr16
	A, [HL]
	A, [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)。其他情况下，Z 标志清零。

[描述示例]

OR A, 0FE98H;　A 寄存器和 FE98H 以位相或，结果存储在 A 寄存器中。

XOR	**异或** **字节数据的逻辑异和**

[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
	A, saddr

助记符	操作数(dst, src)
XOR	A, !addr16
	A, [HL]
	A, [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)的内容以位相异或，结果存储在目的操作数(dst)中。
- 如果逻辑异和显示所有的位是 0，Z 标志被设置为(1)。其他情况下，Z 标志清零。

[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

比较

字节数据的比较

[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
	A, saddr

助记符	操作数(dst, src)
CMP	A, !addr16
	A, [HL]
	A, [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)。
  结果不存储，只有 Z, AC 和 CY 标志改变。
- 如果减法结果是 0，Z 标志被置为(1)。在其他情况下，Z 标志清零(0)。
- 如果减法在第 7 位产生一个借位，CY 标志被置(1)。其他情况下，CY 标志清零。
- 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)。其他情况下，AC 标志清零(0)。

[描述示例]

CMP 0FE38H, #38H; 地址 FE38H 的内容减去 38H，并且只有 Z, AC 和 CY 标志改变(比较地址 FE38H 的内容和立即数)。

5.4 16 位操作指令

如下是 16 位操作指令。

ADDW

加字
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

[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)。其他情况下，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)。其他情况下，CY 标志清零(0)。
- 作为加法的结果，AC 标志未定义。

[描述示例]

ADDW AX, #0ABCDH; ABCDH 和 AX 寄存器的值相加，结果存储在 AX 寄存器中。

SUBW

减字
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

[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)。其他情况下，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)。其他情况下，CY 标志清零(0)。
- 作为减法的结果，AC 标志未定义。

[描述示例]

SUBW AX, #0ABCDH; AX 寄存器的值减去 ABCDH，结果存储在 AX 寄存器中。

CMPW

比较字

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

[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)。其他情况下，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)。其他情况下，CY 标志清零(0)。
- 作为减法的结果，AC 标志未定义。

[描述示例]

CMPW AX, #0ABCDH; AX 寄存器的值减去 ABCDH，并且只有 Z, AC 和 CY 标志改变（比较 AX 寄存器和立即数）。

5.5 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

[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)。其他情况下，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)。其他情况下，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	**减** **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

[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)。其他情况下，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)。其他情况下，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 0FE92H ; 地址 FE92H 的内容递减。

INCW

字加
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

[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

字减
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

[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。

5.6 移位指令

如下是移位指令。

ROR	右移 字节数据右移

[指令格式] ROR dst, cnt

[操作] (CY, $dst_7 \leftarrow dst_0$, $dst_{m-1} \leftarrow dst_m$) × 一次

[操作数]

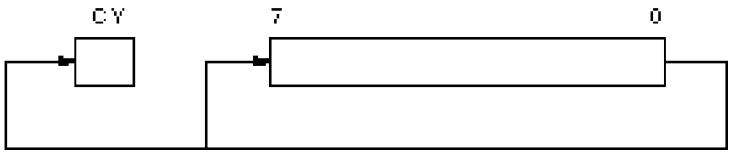
助记符	操作数(dst, cnt)
ROR	A, 1

[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

左移

字节数据左移

[指令格式] ROL dst, cnt

[操作] (CY, $dst_0 \leftarrow 0$, dst_7, $dst_{m+1} \leftarrow dst_m$) × 一次

[操作数]

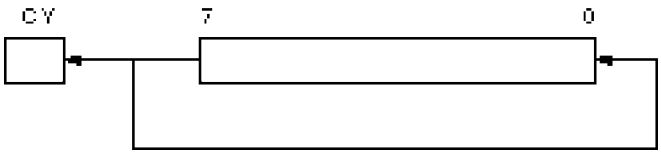
助记符	操作数(dst, cnt)
ROL	A, 1

[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

带进位右移

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) × 一次

[操作数]

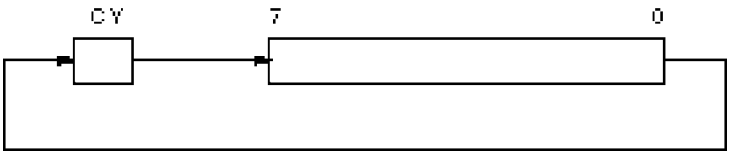
助记符	操作数 (dst, cnt)
RORC	A, 1

[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

带进位左移

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) × 一次

[操作数]

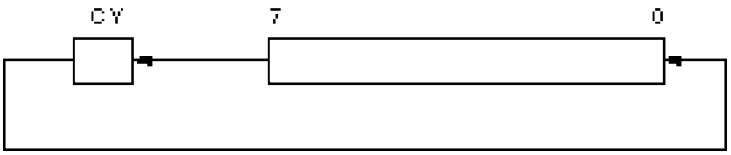
助记符	操作数 (dst, cnt)
ROLC	A, 1

[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 标志右移一位。

5.7 位操作指令

如下是位操作指令。

SET1

设置一位 (进位标志)
1 位数据设置

[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
	PSW.bit
	[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 0FE55H.1; FE55H 的第一位设置为(1)。

CLR1

清一位 (进位标志)
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
	PSW.bit
	[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 Bit 7 位被清零(0)。

NOT1

求反一位 (进位标志)

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。

5.8 调用/返回 指令

如下是 调用/返回 指令。

CALL

调用

子程序调用 (16 位直接调用)

[指令格式] CALL 目标地址

[操作]

(SP – 1) ← (PC + 3)H,
(SP – 2) ← (PC + 3)L,
SP ← SP – 2,
PC ← 目标地址

[操作数]

助记符	操作数 (目标地址)
CALL	!addr16

[标志]

Z	AC	CY

[描述]

- 它是一个通过一个 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 + 3) 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 !3059H; 子程序调用地址 3059H

CALLT

调用表

子程序调用(调用表参考)

[指令格式] CALLT [addr5]

[操作]

(SP – 1) ← (PC + 1)H,
(SP – 2) ← (PC + 1)L,
SP ← SP – 2,
PCH ← (00000000,addr5 + 1)
PCL ← (00000000,addr5)

[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。
- 下一条指令的开始地址(PC + 1) 保存在堆栈内，并且跳转到调用表内的字数据(地址的高 8 位固定为 00000000B，并且接下来的 5 位由 addr5 指定)指定的地址处执行。

[描述示例]

CALLT [40H]; 子程序调用，跳转到 0040H 和 0041H 指向的字数据地址处执行。

RET	**返回** **从子程序返回**

[指令格式] RET

[操作]

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

[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

从中断返回
从硬件向量中断返回

[指令格式] RETI

[操作]

$PC_L \leftarrow (SP)$,
$PC_H \leftarrow (SP + 1)$,
PSW $\leftarrow$ (SP + 2),
SP $\leftarrow$ SP + 3,
NMIS $\leftarrow$ 0

[操作数]

无

[标志]

Z	AC	CY
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。
- 在该指令和下一条指令执行中间，不会响应任何中断。
- NMIS 标志在响应一个不可屏蔽中断时被设置为 1，随后被 RETI 指令清零。

[注意事项]

当从不可屏蔽中断服务程序返回的执行不是通过 RETI 指令时，NMIS 标志不被清零，所以不能响应任何中断(包括不可屏蔽中断)。

5.9 堆栈操作指令

如下是堆栈操作指令。

PUSH

压栈
压栈

[指令格式] PUSH src

[操作]

当 src = rp

(SP – 1) ← src_H,
(SP – 2) ← src_L,
SP ←SP – 2

当 src = PSW

(SP – 1) ← src
SP ← SP – 1

[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

出栈

出栈

[指令格式] POP dst

[操作]

当 dst = rp	当 dst = PSW
dst_L ← (SP),	dst ← (SP)
dst_H ← (SP + 1),	SP ← SP + 1
SP ← SP + 2	

[操作数]

助记符	操作数 (dst)
POP	PSW
	rp

[标志]

dst = rp

Z	AC	CY

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, AX
MOVW AX, SP

移动字

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, AX
	AX, 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, AX; AX 寄存器内容保存在堆栈指针中。

5.10 无条件跳转指令

如下是一个无条件跳转指令。

```
BR ... 102
```

BR	**跳转** **无条件跳转**

[指令格式] BR 目的地址

[操作] PC ← 目的地址

[操作数]

助记符	操作数 (目的地址)
BR	!addr16
	AX
	$addr16

[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 AX; AX 寄存器内容被认为是程序跳转的地址。

5.11 条件跳转指令

如下是条件跳转指令。

BC	**如果进位跳转** **带进位标志 (CY = 1)的条件跳转**

[指令格式] BC $addr16

[操作] PC ← PC + 2 + jdisp8 如果 CY = 1

[操作数]

助记符	操作数 ($addr16)
BC	$addr16

[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 $300H; 当 CY = 1，程序跳转到 0300H (该指令的开始地址的地址范围从 027FH ~ 037EH)。

BNC

如果无进位跳转
带进位标志 (CY = 0)的条件跳转

[指令格式] BNC $addr16

[操作] PC ← PC + 2 + jdisp8 if CY = 0

[操作数]

助记符	操作数 ($addr16)
BNC	$addr16

[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 $300H; 当 CY = 0，程序跳转到 0300H (该指令的开始地址的地址范围从 027FH ~ 037EH)。

BZ

如果是零跳转
带零标志 (Z = 1)的条件跳转

[指令格式] BZ $addr16

[操作] PC ← PC + 2 + jdisp8 如果 Z = 1

[操作数]

助记符	操作数 ($addr16)
BZ	$addr16

[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 $3C5H; 当 B 寄存器是 0，程序跳转到 03C5H (该指令的开始地址的地址范围从 0344H ~ 0443H)。

BNZ

如果是非零跳转
带零标志 (Z = 0)的条件跳转

[指令格式] BNZ $addr16

[操作] PC ← PC + 2 + jdisp8 如果 Z = 0

[操作数]

助记符	操作数 ($addr16)
BNZ	$addr16

[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 $0A39H; 如果 A 寄存器不是 0055H，程序跳转到 0A39H (该指令的开始地址的地址范围从 09B8H ~ 0AB7H)。

BT

如果真跳转

带位测试 (字节数据位 = 1)的条件跳转

[指令格式] BT bit, $addr16

[操作] PC ← PC + b + jdisp8 如果 bit = 1

[操作数]

助记符	操作数 (bit, $addr16)	b (字节数)
BT	saddr.bit, $addr16	4
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	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($addr16)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 0FE47H.3, $55CH; 当 FE47H 的第 3 位是 1，程序跳转到 055CH (该指令的开始地址的地址范围从 04DAH ~ 05D9H)处。

BF

如果假跳转

带位测试 (字节数据位 = 0)的条件跳转

[指令格式] BF bit, $addr16

[操作] PC ← PC + b + jdisp8 如果 bit = 0

[操作数]

助记符	操作数 (bit, $addr16)	b (字节数)
BF	saddr.bit, $addr16	4
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	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($addr16)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, $1549H; 当端口 2 的第 2 位是 0，程序跳转到 1549H (该指令的开始地址的地址范围从 14C6H ~ 15C5H)处。

DBNZ

递减并且如果不是零跳转条件循环 (R1 ≠ 0)

[指令格式] DBNZ dst, $addr16

[操作] dst ← dst – 1,
then PC ← PC + b + jdisp16 如果 dst R1 ≠ 0

[操作数]

助记符	操作数 (dst, $addr16)	b (字节数)
DBNZ	B, $addr16	2
	C, $addr16	2
	saddr, $addr16	3

[标志]

Z	AC	CY

[描述]

- 从第一个操作数指定的目的操作数 (dst) 减一，并把结果存储在目的操作数(dst)中。
- 如果减的结果不是 0，程序跳转到第二个操作数 ($addr16)指定的地址处。
  如果减的结果是 0，不执行任何操作然后执行下一条指令。
- 标志的值保留。

[描述示例]

DBNZ B, $1215H; B 寄存器递减。如果结果不是 0，程序跳转到 1215H (该指令的开始地址的地址范围从 1194H ~ 1293H)处。

5.12 CPU 控制指令

如下是 CPU 控制指令。

NOP	无操作 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	**允许中断** **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))。
- 在该指令执行后，中断被立即响应。
- 如果执行该指令，其他源的向量中断响应可被禁止。如需了解详细信息，每个产品用户手册的 **"中断功能"。**

DI	**禁止中断** **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))。
- 没有中断在该指令和下一条指令之间被响应。
- 如需了解中断服务程序的详细信息，每个产品用户手册的 **"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。功率耗散会被最小化到一个仅有非常低的电流水平。

附录 A 指令索引(助记符: 按功能)

[备忘录]

附录 B 指令索引(助记符: 按字母顺序)

[备忘录]

附录 C 修订历史

此版本的修订历史显示如下。章节表示各个版本的章节。

版本	内容	章节
2nd	增加以下产品 μPD789026, 789407, 789417, 789800, 和 789806Y 子系列	全部
	修改 78K/0S 系列产品内部数据存储空间表的格式	**第一章 存储器空间**
3rd	增加以下产品 μPD789046, 789104, 789114, 789124, 789134, 789146, 789156, 789167, 789177, 789197AY, 789217AY, 789407A, 789417A, 和 789842 子系列	全部
	删除以下产品 μPD789407, 789417, 和 789806Y 子系列	
	修改 MOV PSW, #byte 指令码	**第四章 指令集**
	修改 MOVW rp, AX 指令码	
	修改 XOR A, r 指令码	
	修改 CMP A, r 指令码	

[备忘录]

详细信息请联系：

（中国区）

网址：
http://www.cn.necel.com/
http://www.necel.com/

[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