iMQ
Your First and Best MCU

i87 指令集中文版

V1.0

目录

1. 修改记录

Version	Approved Date	Description	Issuer
V1.0	2011/12/18	新建立	陈慧遂 许平育

2. 概观

i87 系列单片机具有 133 种类型的数据操作指令，提供 732 条指令。下表列出按类型分类的所有指令。指令长度为 1 至 5 字符，最常用到的指令采用短的指令长度，以增加建立较节省内存的程序。i87 提供一个基于存储映像输出/输入系统的简单指令架构。虽然只有 42 种助记符，但可支持 8 种强大存储器操作的寻址模式。

除了基本的机器指令之外，另外 i87 还有提供有助于编程方便的汇编扩展指令。

传送交换	8 位数据传送、交换		7 种	49 条
	16 位数据传送、交换		7	43
	标志操作		5	5
	堆栈指针操作		1	2
	压栈，弹出		4	6
数据操作	8 位操作	比较	4	29
		增量(+1)、减量(-1)	4	28
		算术运算	16	116
		逻辑运算	12	87
		十进制调整	2	2
	16 位操作	比较	3	15
		增量(+1)、减量(-1)	2	2
		算术运算	12	60
		逻辑运算	9	45
		二补数	1	1
	乘除法		2	2
移位循环	移位	8 位(逻辑)	2	2
		16 位(算数)	2	2
	循环	8 位	2	2
半字节操作	半字节交换、半字节循环		3	27
位操作	置"1"、清"0"、补数、传送、交换、异或		18	162
分支	跳转		6	24
	调用		4	16
	返回		3	3
其他	软中断、其他		2	2
总计			133 种	732 条

表 2.1 i87 指令集

寄存器间接	7 种
直接	2
寄存器	1
立即	1
相关	2
绝对	1
向量	1
直接位	2
寄存器间接位	1
总计	18

表 2.2 i87 寻址方式

2.1 符号说明

下面是指令、寻址方式中所用到的符号说明。

符号	说明	符号	说明
A	A 寄存器	r, g	8 位寄存器(参阅表 1.3)
W	W 寄存器	rr, gg	16 位寄存器(参阅表 1.4)
B	B 寄存器	n	4 位或 8 位立即数
C	C 寄存器	mn	16 位立即数
D	D 寄存器	d	5 位或 8 位带符号偏移量(-16 到+15 或-128 到+127)
E	E 寄存器	x, y	8 位直接地址(0000H 到 00FFH)
H	H 寄存器	vw, uz	16 位直接地址(0000H 到 FFFFH)
L	L 寄存器	(XX)	被有效地址 XX 标志的内存地址中的内容
WA	WA 寄存器	(XX + 1, XX)	由 XX 标识地址 ·在内存中两个连续字的内容
BC	BC 寄存器	b	位(0 到 7)
DE	DE 寄存器	.b	由 b 标识的位内容
HL	HL 寄存器	←	传送
IX	IX 寄存器	⇔	交换
IY	IY 寄存器	+	加
PC	程序计数器	-	减
SP	堆栈指针	×	乘
PSW	程序状态字	÷	除

符号	说明	符号	说明
JF	跳转状态标志位	&	按位逻辑与
ZF	零标志位	|	按位逻辑或
CF	进位标志位(1 位累加器)	^	按位逻辑异或
HF	半进位标志位	null	空操作
SF	带符号标志	$	本指令起始地址(程序计数器指向在$之后的 2 或 3 个字节)
VF	溢值标志位	(src)	源内存
CF 非	进位标志位取反	(dst)	目标内存
IMF	中断控制使能标志位	RBS	寄存器组选择器
NxtOp	下一个操作地址(下一个指令的开始地址)		

r, g	8 位寄存器
0	A
1	W
2	C
3	B
4	E
5	D
6	L
7	H

表 2.3

rr, gg	16 位寄存器
0	WA
1	BC
2	DE
3	HL
4	IX
5	IY
6	SP
7	HL

表 2.4

标志位设定条件	
*	由操作标志的数据设置
Z	零检测标志位设置 • 传送 当 8 位数据为"00H"时，设为"1" ，否则清除为"0"。 • 交换 当 g 寄存器或(src)在交换之前为"00H"时，设为"1" ，否则设为"0"。 • ALU 当操作结果为"00H"8 位操作)或"0000H"(16 位操作)时，设为"1"，否则清除为"0"。然而，在乘法中乘积的高 8 位或除法中的余数为"00H"时，设为"1"，否则清除为"0"。 • 移位循环 当寄存器经移位或循环操作后结果为"00H"时，设为"1" ，否则清除为"0"。 • 其他 当 Z 在 JF 栏出现时，代表在 ZF 中的资料同样设置到 JF 中。
C	进位标志位设置 • 加法 根据最高位的进位设置 • 减法 根据最高位的借位设置 • 除法 当除数是"00H"，或商大于或等于"100H"时，设为"1"，否则清除为"0"。 • 其他 当 C 在 JF 栏出现时，代表在 CF 中的资料同样设置到 JF 中。
$\overline{C}$	将 CF 中的资料取反
H	半进位标志位设置 • 加法 根据第三位进位设置 • 减法 根据第三位借位设置
S	带符号(数据中最重要的位)
V	溢值
$\overline{J}$	将 JF 中的资料取反
1	总是为"1"
0	总是为 "0"
U	没有资料设置
-	标志位无效，保持指令执行前的数据

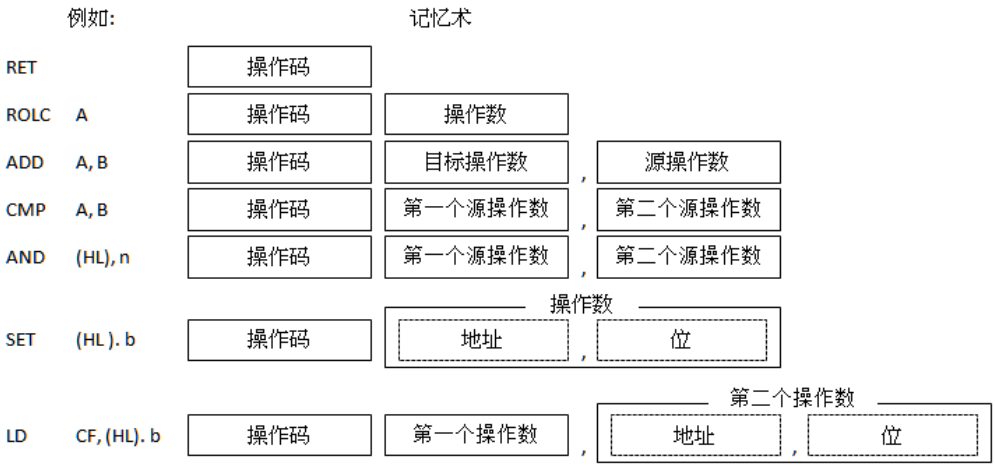
2.2 指令助记符

下面描述 i87 系列指令的助记符规则。

每一个助记符包含一个操作码和一个操作数(部分指令并没有操作数)。操作码和操作术之间以一个以上的空白分隔。如果一个指令包含两个以上的操作数，则每一个操作数之间用逗号分隔。

如果指令中的操作数包含源内存和目标内存，目标内存永远放在源内存之前；如果有多个源内存，则第一个源内存放在第一个位置，然后其他源内存依序放在第一个源内存之后。

如果操作数中包含位标识域，则地址和位标识域之间用句点分隔。



助记符	说明
ADD	加法
ADDC	带进位加法
AND	逻辑与
CALL	调用
CALLV	向量调用
CLR	清字节/位
CMP	比较
CPL	一位求反
DAA	十进制判断 8 位加法
DAS	十进制判断 8 位减法
DEC	减 1 字节/字(寄存器)
DI*	可屏蔽中断关
DIV	除法
EI*	中断开
INC	加 1 字节/字(寄存器)
J*	无条件跳转
JP	绝对跳转
JR	相对跳转
JRS	短相对跳转
LD	位/字节/字数据加载
LDW	字载入
MUL	乘法
NEG	负值
NOP	空操作
OR	逻辑或
POP PUSH	弹出 堆栈

助记符	说明
RET	从子程序返回
RETI	从可屏蔽中断子程序返回
RETN	从不可屏蔽中断子程序返回
ROLC	带进位左循环
ROLD	左循环
RORC	带进位右循环
RORD	右循环
SET	位设置
SHLC	逻辑带进位左移
SHLCA	演算带进位左移
SHRC	逻辑带进位右移
SHRCA	演算带进位右移
SUB	减法
SUBB	带借位减
SWAP	半字节交换
SWI	软件中断
TEST*	位检测
XCH	交换
XOR	逻辑异或

表 2.5

注：“*”代表汇编扩展机器指令

2.3 目标代码格式

i87 系列的指令有一个字节和两个字节的操作码。

2.3.1 一个字节的操作码格式

操作码放在第一个字节，操作数放在第二个字节以及接下来的字节。如果操作数有两个字节，低位字节则放在高位字节的前面。当操作数包含源和目标时，如果源内存是立即数的话，源内存则放在目标内存的后面，像是 LD(x), n。

例如:		第一个字节	第二个字节	第三个字节	第四个字节
LD	A, B	操作码			
LD	A, n	操作码	n		
LD	WA, mn	操作码	n	m	
LD	(x), n	操作码	x	n	
LDW	(x), mn	操作码	x	n	m

2.3.2 两个字节的操作码格式

第一个操作码放在第一个字节，第一个字节定义的操作数放在下一个的字节。然后第二个字节的操作码和第二个字节操作码定义的操作数放在接下来的字节。

如果第一个操作码是用来定义寻址模式的话，则这一个操作码称为前缀码。前缀码有两种，一种是定义寄存器的寄存器前缀码，另外一种是定义源内存或目标内存的内存前缀码。注意下列五个指定的第一个操作码忽略了特定寄存器的内容。

(1) RETN
(2) LD PSW, n
(3) PUSH PSW
(4) POP PSW
(5) JR M/P/SLT/SGE/SLE/SGT/VS/VC, a

例如:		第一个字节	第二个字节	第三个字节	第四个字节
LD	B, C	第一个操作码	第二个操作码		
ADD	B, n	第一个操作码	第二个操作码	n	
LD	A, B	第一个操作码	第二个操作码		
LD	WA, mn	第一个操作码	x	第二个操作码	
LD	(x), n	第一个操作码	d	第二个操作码	
LDW	(HL+d), n	第一个操作码	d	第二个操作码	n

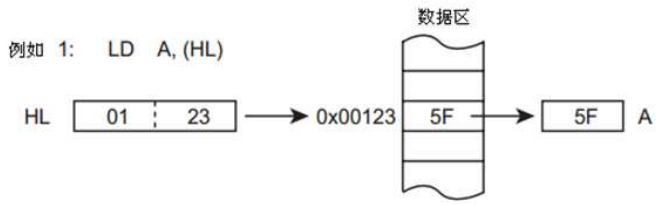
2.4 寻址方式

i87 系列支持 17 种寻址模式。某些指令使用了多种寻址模式的组合。

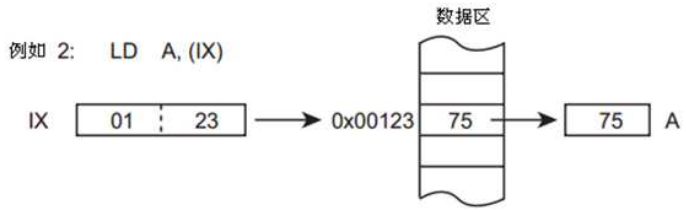
2.4.1 寄存器间接寻址

2.4.1.1 寄存器间接寻址 HL、DE、IX、IY

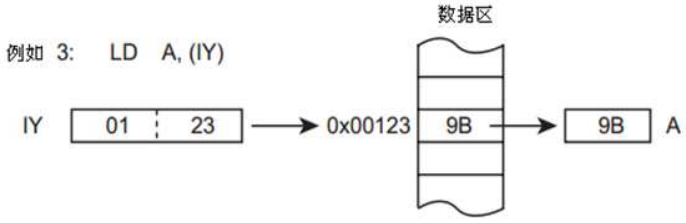
有效地址由 16 位寄存器 HL、DE、IX、IY 的内容决定。



例 1 说明：HL 寄存器的内容 0123H 直接指定有效存储器地址为 0123H，地址 0123H 单元中的内容 5FH 传送至累加器 A 中。



例 2 说明：IX 寄存器的内容 0123H 直接指定有效存储器地址为 0123H，地址 0123H 单元中的内容 75H 传送至累加器 A 中。

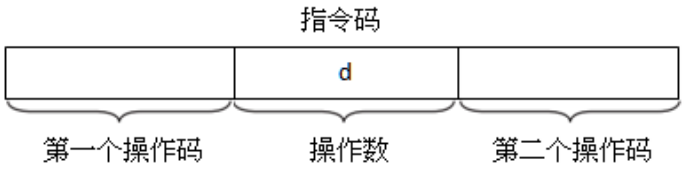


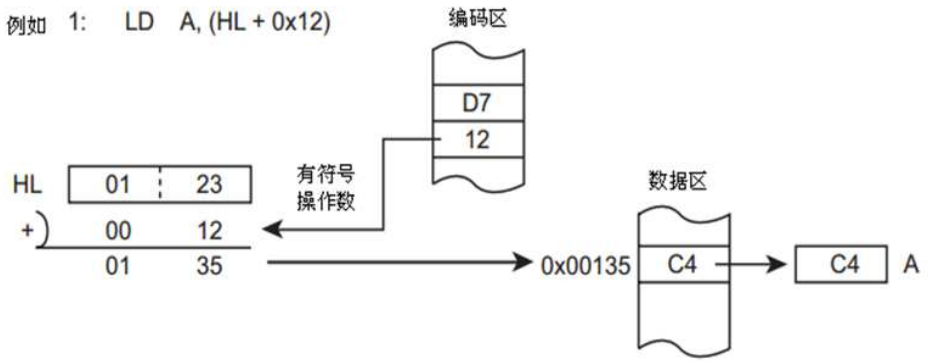
例 3 说明：IY 寄存器对的内容 0123H 直接指定有效存储器地址为 0123H，地址 0123H 单元中的内容 9BH 传送至累加器 A 中。

2.4.1.2 8 位转移间接偏址寻址 (HL + d)、(IX + d)、(IY + d)

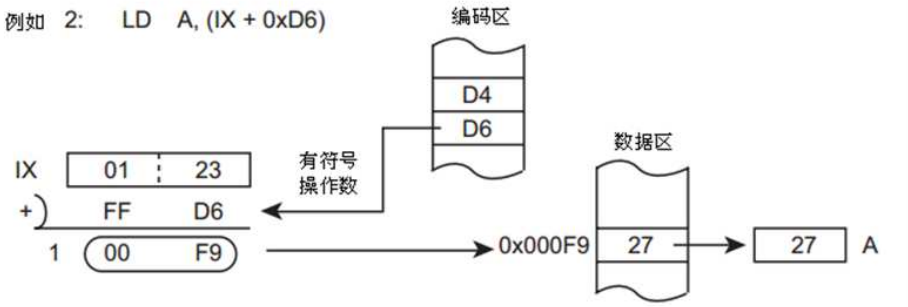
有效地址由 16 位寄存器 HL、IX、IY 与 8 位转移量 d 的有符号操作数(如下表所示)的和决定。数据传送后，16 位寄存器不发生变化。

8位转移量d	有符号操作数
00H ~ 7FH	0000H ~ 007FH (0 ~ +127)
80H ~ FFH	FF80H ~ FFFFH (-128 ~ -1)





例 1 说明：HL 寄存器的内容 0123H 加上转移量 12H 的有符号操作数 12H 后，直接指定有效存储器地址为 0135H；地址 0135H 单元中的内容 C4H 传送至累加器 A 中。HL 寄存器的内容 0123H 和地址 0135H 单元中的内容 C4H 不变。

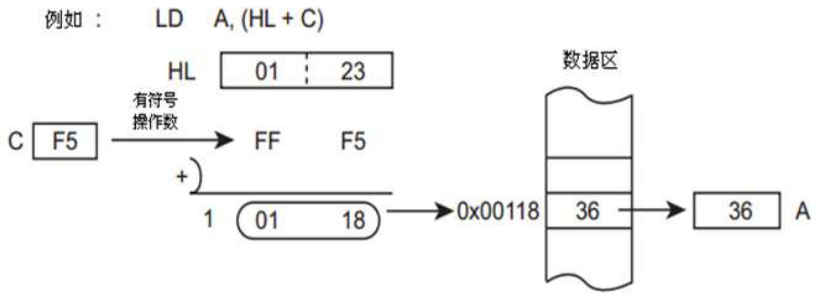


例 2 说明：IX 寄存器的内容 0123H 加上转移量 D6H 的有符号操作数 FFD6H 后，直接指定有效存储器地址为 00F9H；地址 00F9H 单元中的内容 27H 传送至累加器 A 中。IX 寄存器的内容 0123H 和地址 00F9H 单元中的内容 27H 不变。

2.4.1.3 寄存器偏址寻址 HL + C

有效地址由 HL 寄存器与寄存器 C 的有符号操作数(如下表所示)的和决定。数据传送后，HL 寄存器和 C 寄存器不发生变化。

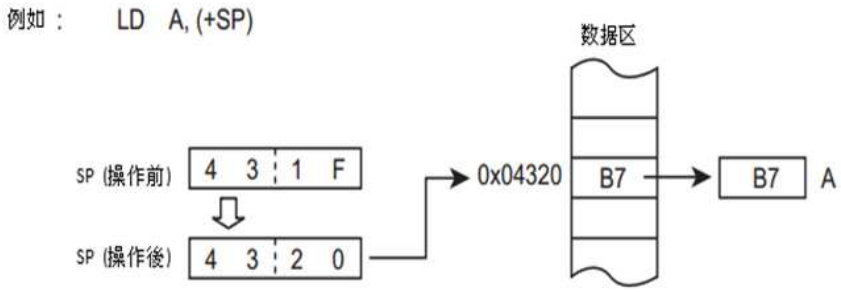
寄存器 C	有符号操作数
00H ~ 7FH	0000H ~ 007FH (0 ~ +127)
80H ~ FFH	FF80H ~ FFFFH (-128 ~ -1)



说明：HL 寄存器的内容 0123H 加上 C 寄存器的有符号操作数 FFF5H 后，直接指定有效存储器地址为 0118H；地址 0118H 单元中的内容 36H 传送至累加器 A 中。HL 寄存器的内容 0123H、地址 0118H 单元中的内容 36H 和 C 寄存器的内容 F5H 不变。

2.4.1.4 堆栈指针间接寻址，SP 前加 (+SP)

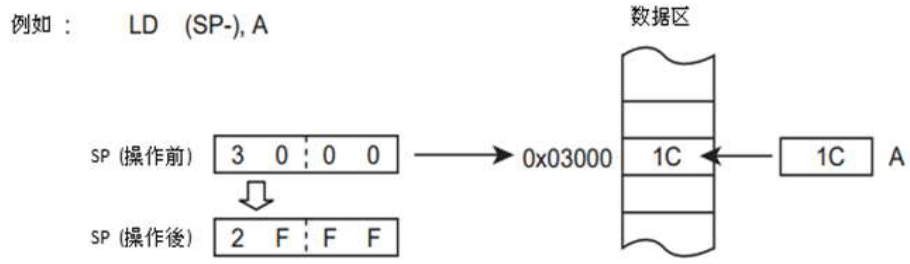
首先堆栈指针自动加 1，有效地址由加后的堆栈指针决定，而标志位没有任何变化。这个寻址模式只适用于源程序内存地址的操作。



说明：堆栈指针的内容 431FH 自动加 1，自动加 1 的内容直接指定有效存储器地址为 4320H；地址 4320H 单元中的内容 B7H 传送至累加器 A 中。地址 4320H 单元中的内容 B7H 不变。

2.4.1.5 堆栈指针间接寻址，SP 后减 (SP-)

首先堆栈指针自动加 1，有效地址由加后的堆栈指针决定，而标志位没有任何变化。这个寻址模式只适用于源程序内存地址的操作。

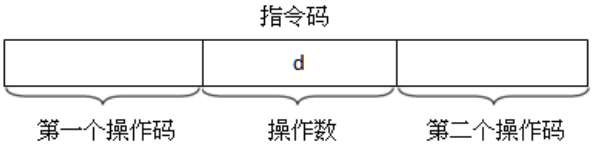


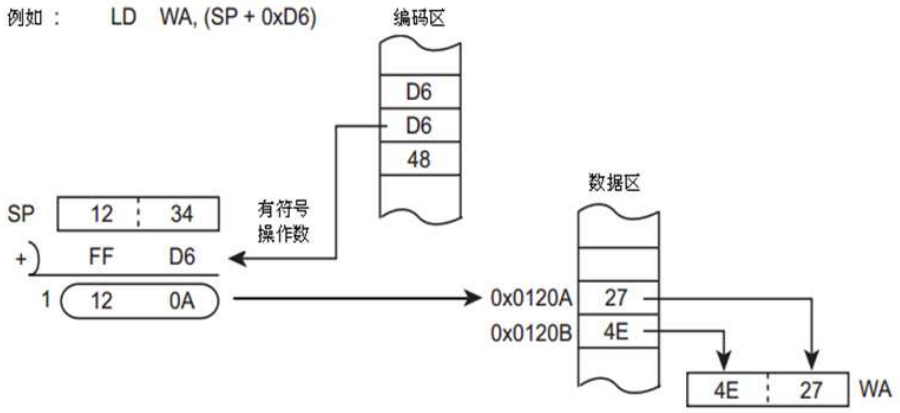
说明:堆栈指针的内容 3000H 直接指定有效存储器地址为 3000H ;地址 3000H 单元中的内容 1CH 传送至累加器 A 中后，堆栈指针自动减 1。地址 3000H 单元中的内容 1CH 不变。

2.4.1.6 堆栈指针 8 位转移量间接寻址 SP + d

有效地址由堆栈指针与 8 位转移量 d 的有符号操作数(如下表所示)的和决定。

8位转移量 d	有符号操作数
00H ~ 7FH	0000H ~ 007FH (0 ~ +127)
80H ~ FFH	FF80H ~ FFFFH (-128 ~ -1)



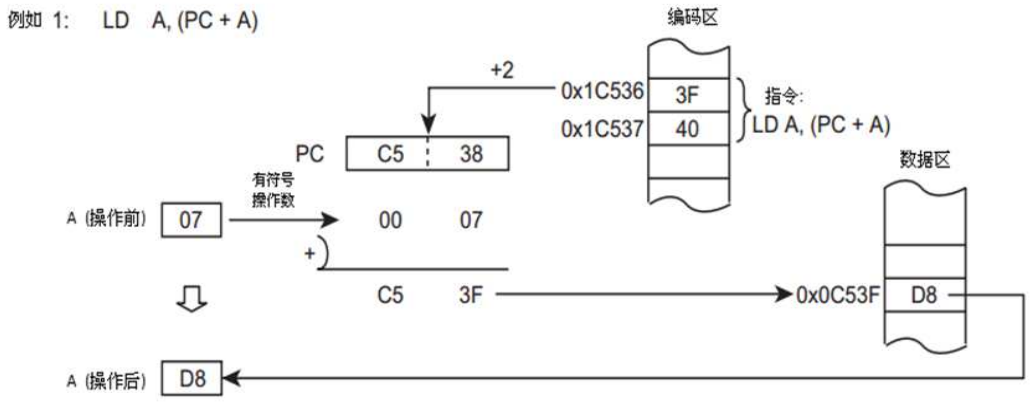


说明：堆栈指针 SP 的内容 0123H 加上 8 位转移量 D6H 的有符号操作数 FFD6H 后，直接指定有效存储器地址为 0120AH；由地址 0120A H 单元开始的连续两个地址中的内容传送至寄存器 WA 中。

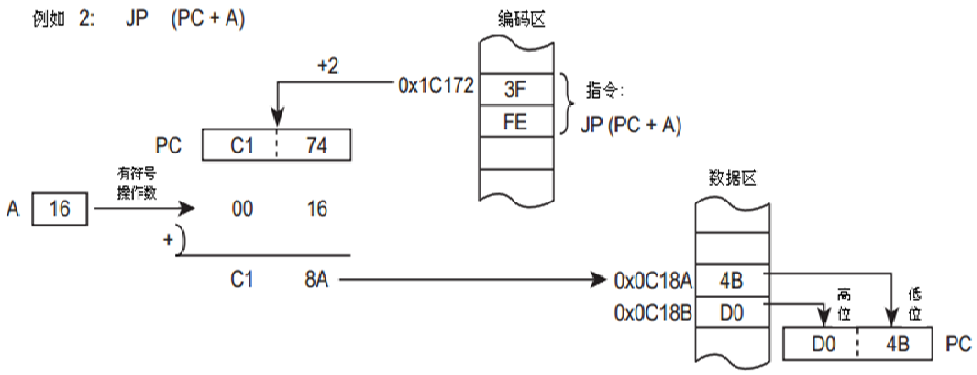
2.4.1.7 寄存器相关偏址寻址 PC + A

有效地址由程序计数器的当前内容(执行指令的头地址加 2)与累加器 A 的有符号操作数(如下表所示)的和决定。这种寻址方式只能用于源程序内存地址的操作，它可以使代码转换，例如 BCD 码转换成 7 段显示码、查表等变得更容易。

累加器 A	有符号操作数
00H ~ 7FH	0000H ~ 007FH (0 ~ +127)
80H ~ FFH	FF80H ~ FFFFH (-128 ~ -1)



例 1 说明：本指令起始地址 C172H 加上 2 后为 C174H，传送至程序计数器。程序计数器的内容 C174H 加上累加器 A 的内容 16H 的有符号操作数 0016H 后，直接指定有效存储器地址为 C18AH；地址 C18AH 单元中的内容 4BH 传送至程序计数器低 8 位，地址 C18BH 单元中的内容 D0H 传送至程序计数器高 8 位。程序从 D04BH 地址单元开始执行，累加器 A 的内容 16H、地址 C18A 单元中的内容 4BH 和地址 C18B 单元中的内容 D0H 不变。

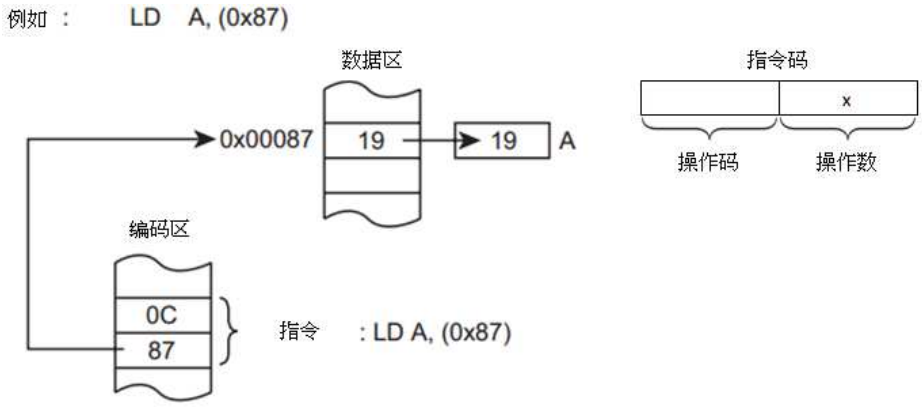


例 2 说明：本指令起始地址 C172H 加上 2 后为 C174H，传送至程序计数器。程序计数器的内容 C174H 加上累加器 A 的内容 16H 的有符号操作数 0016H 后，直接指定有效存储器地址为 C18AH；地址 C18AH 单元中的内容 4BH 传送至程序计数器低 8 位，地址 C18BH 单元中的内容 D0H 传送至程序计数器高 8 位。程序从 D04BH 地址单元开始执行，累加器 A 的内容 16H、地址 C18A 单元中的内容 4BH 和地址 C18B 单元中的内容 D0H 不变。

2.4.2 直接寻址

2.4.2.1 8 位直接寻址 (x)

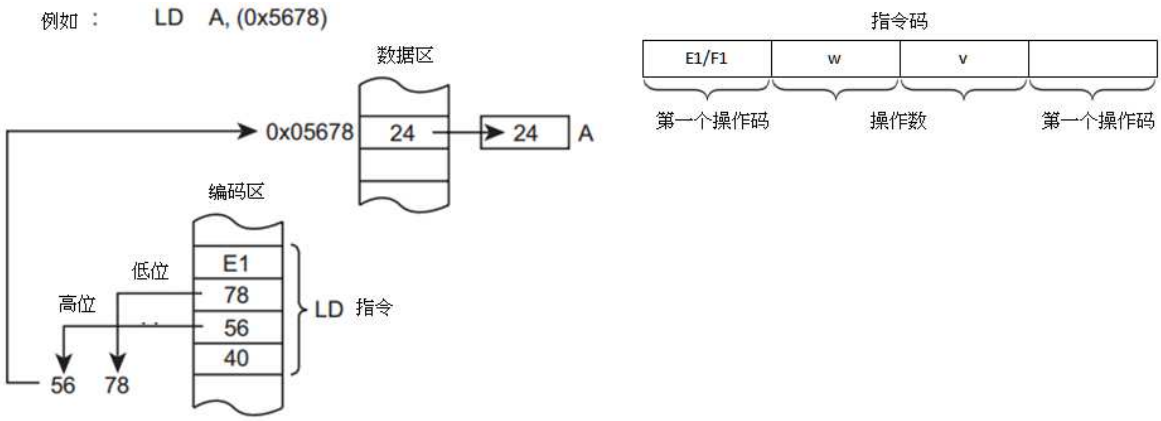
有效地址由目标码中的 8 位数据 x 直接标识。这种寻址方式只允许在内存的最低 256 个字节内 (00000H ~ 000FFH)。



说明：目标码 8 位操作数 x 直接指定的有效存储器地址 0087H；地址 0087H 单元中的内容 19H 传送至累加器 A 中。地址 0087H 单元中的内容 D8H 不变。

2.4.2.2 16 位直接寻址 (vw)

有效地址由目标码中的 16 位数 vw 直接标识。这种寻址方式只允许在内存的最低 256 个字节内 (00000H ~ 0FFFFH)。



例 1 说明：目标码 16 位操作数 vw 直接指定的有效存储器地址 5678H；地址 5678H 单元中的内容 24H 传送至累加器 A 中。地址 5678H 单元中的内容 24H 不变。

2.4.3 寄存器寻址 (r 或 rr)

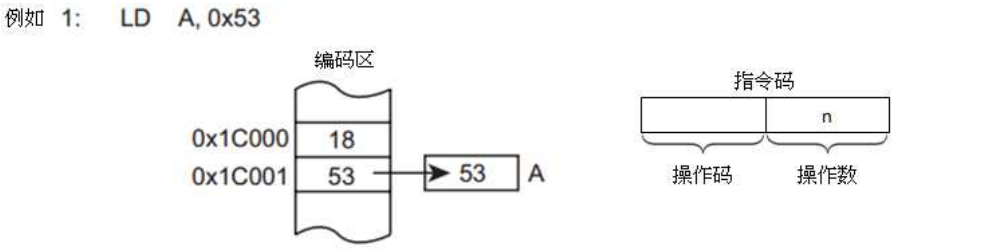
把由寄存器标识域中的操作码所标识的寄存器变为操作对象。

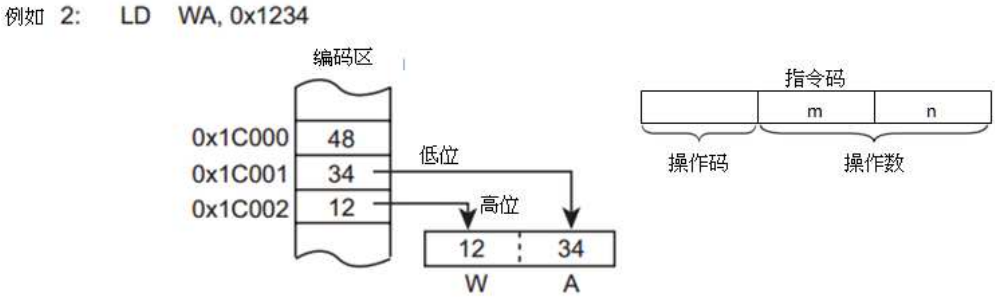
例 1 说明：被寻址的累加器 A 为目标操作数，寄存器 B 为源操作数。指令执行后，寄存器 B 的内容传送至累加器 A 中，寄存器 B 中的内容不变。

例 2 说明：被寻址的寄存器 DE 为目标操作数。指令执行后，寄存器 DE 的内容加 1。

2.4.4 立即寻址(n 或 nm)

在目标码中的立即数成为操作对象。如果是 16 位的立即数，则操作时将对应于低 8 位和高 8 位的顺序。立即数不要在括号中，否则在汇编语言中将当作直接地址寻址方式。





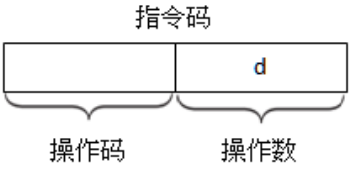
2.4.5 相关寻址

在目标码中的立即数成为操作对象。如果是 16 位的立即数，则操作时将对应于低 8 位和高 8 位的顺序。立即数不要在括号中，否则在汇编语言中将当作直接地址寻址方式。

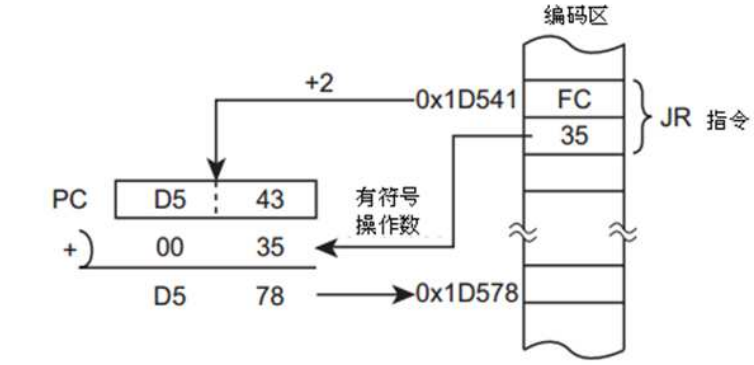
2.4.5.1 8 位转移 PC 相关寻址

有效地址由操作码中 8 位转移量 d 的有符号操作数(如下表)与程序计数器(执行指令的头地址加 2 或加 3)的当前值相加决定。

8位转移量d	有符号操作数
00H ~ 7FH	0000H ~ 007FH (0 ~ +127)
80H ~ FFH	FF80H ~ FFFFH (-128 ~ -1)

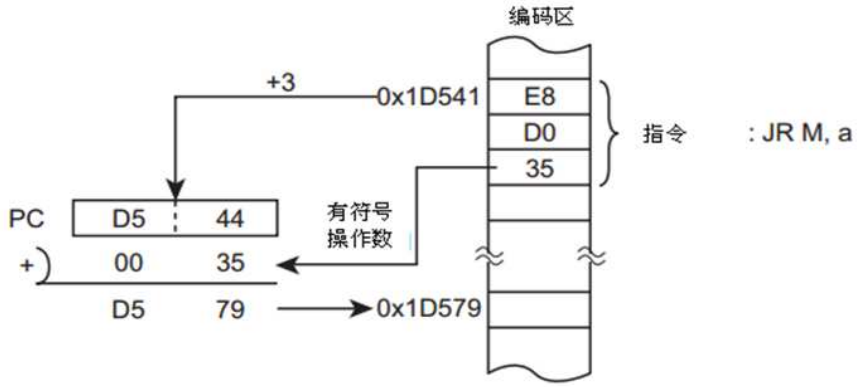


例如 1: JR $ + 2 + 35H 或 JR 0x0D578



例 1 说明 : 本指令的起始地址$(1D541H)加上 2 为程序计数器的内容，程序计数器的内容 1D543H 加上 8 位转移量 0035H 的有符号操作数 0035H 后，直接指定有效储存器地址 1D578H，程序在 1D578H 地址单元开始执行。

例如 2: JR M, $ + 3 + 35H 或 JR M, 0x0D579

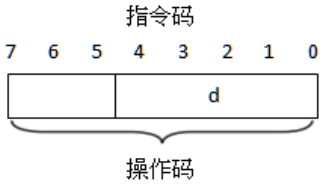


例 2 说明：本指令的起始地址$(1D541H)加上 3 为程序计数器的内容，程序计数器的内容 1D544H 加上 8 位转移量 0035H 的有符号操作数 0035H 后，直接指定有效储存器地址 1D579H(如果跳转状态标志位设为" 1" 的话)，程序在 1D579H 地址单元开始执行。

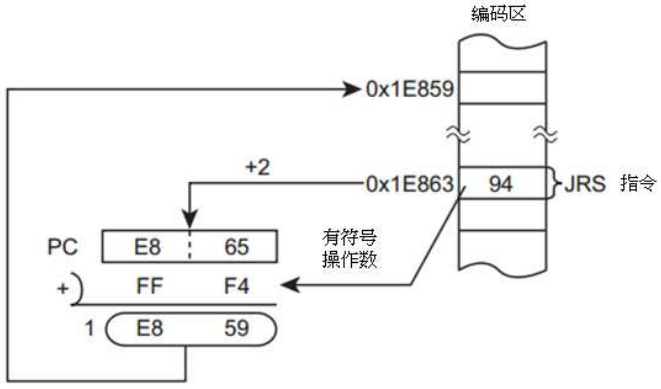
2.4.5.2 5 位转移 PC 相关寻址

有效地址由操作码中 5 位转移量 d 的有符号操作数(如下表)与程序计数器(执行指令的头地址加 2)的当前值相加决定。

5位转移量d	有符号操作数
00H ~ 0FH	0000H ~ 000FH (0 ~ +15)
10H ~ 1FH	FF00H ~ FFFFH (-16 ~ -1)



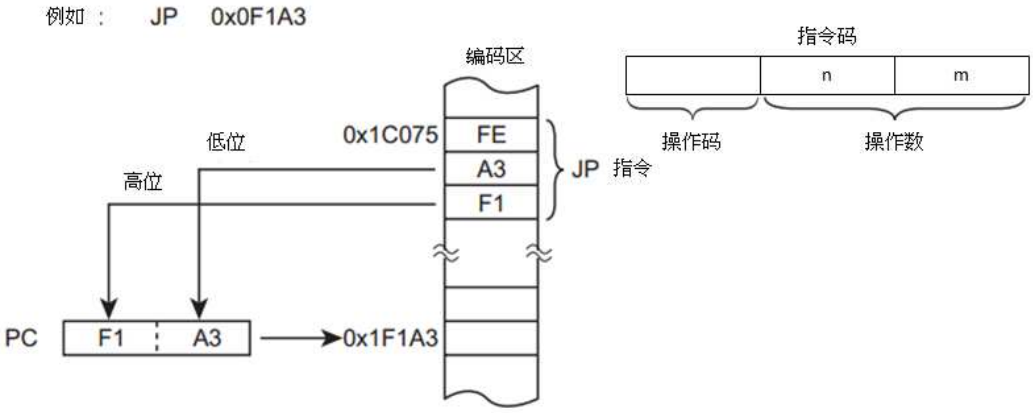
例如： JRS T, $ + 2 + 14H 或 JRS T, 0x0E859



说明：本指令起始地址$ (E1863H) 加上 2，再加上 FFF4H (14H 转移量的有符号操作数) 则为 PC 寄存器的内容 1E859H (如果跳转状态标志位设为" 1" 的话)，程序从 1E859H 开始执行。

2.4.6 绝对寻址

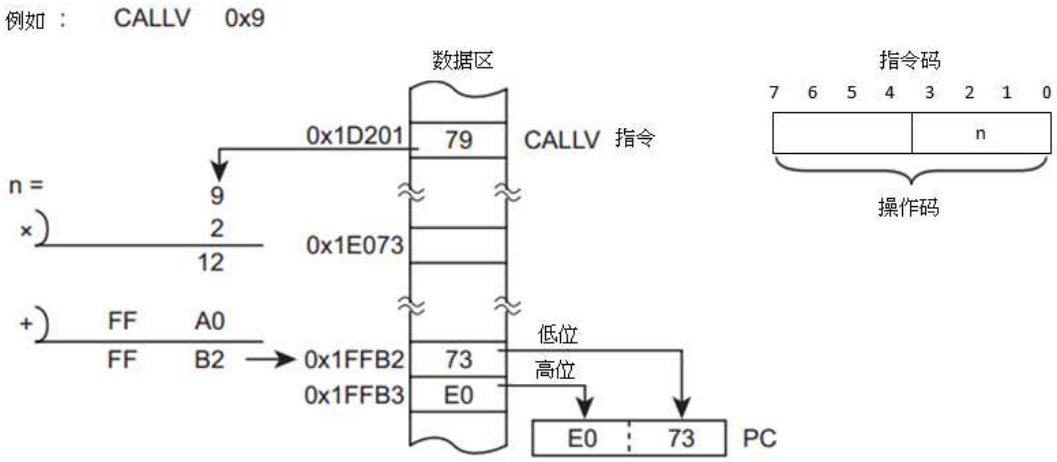
有效地址由目标码中 16 位数据标识。



说明：目标码 16 位操作数 F1A3H 直接指定有效储存器地址 F1A3H，F1A3H 传送到 PC 寄存器，程序从 F1A3H 开始执行。

2.4.7 向量寻址

有效地址由程序内存中读出的 16 位数据(向量地址)标识，地址是 4 位数据 n 的两倍加上 1FFA0H。



说明：目标码低 4 位操作数 9H 乘以 2 后为 12H，加上 1FFA0H 变成 1FFB2。地址 1FFB2 中的内容 73H 传送到 PC 寄存器低 8 位，地址 1FFB3 中的内容 E0H 传送到 PC 寄存器高 8 位，PC 寄存器的内容则为 E073H。

2.4.8 直接位寻址

2.4.8.1 寄存器位寻址

把操作码中寄存器标识域和位标识域标识的寄存器位变为操作对象。

例如： SET A.3

A寄存器的第3位设为"1"

2.4.8.2 内存直接位寻址

把直接寻址方式，例如(HL)、(DE)、(IX)、(IY)、(HL + d)、(IX + d)、(IY + d)、(HL + C)、(+SP)、(SP + d)、(PC + A)、(x)或(vw)所标识内存地址的操作码的位标识域作为直接操作对象。

例如 1: SET (HL). 1

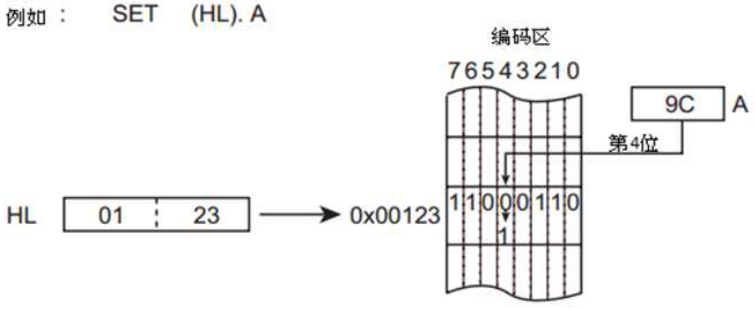
例如 2: SET (HL + 0x57). 6

例如 3: SET (0x00058). 3

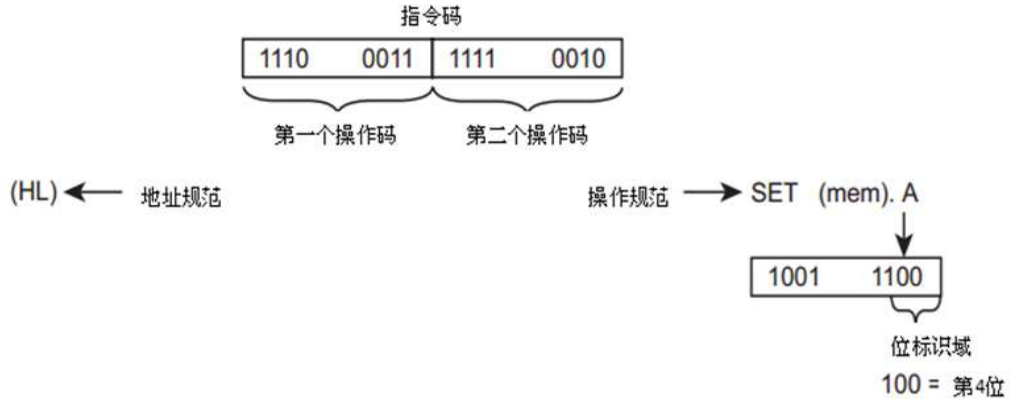
2.4.9 寄存器间接位寻址

把寄存器对(HL)、(DE)、(IX)、(IY)、(HL + d)、(IX + d)、(IY + d)、(HL + C)、(+SP)、(SP + d)、(PC + A)、(x)或(vw)中对应的内存地址的操作码中寄存器标识标识的 8 位寄存器的低 3 位的内容作为直接操作对象。

例如： SET (HL). A



说明：累加器 A 的内容 9CH 低 3 位数直接指定第 4 位。HL 寄存器地址 0123H 中的内容 C6H 的第 4 位设为" 1" 后变成 D6H。



3. 汇编指令应用举例

3.1 数据传送、交换指令

3.1.1 LD A, (x)

本命令将 8 位数据从储存器装载到寄存器具有零测试功能。例如：命令〔LD A, (x)〕从内存地址 x 读数据到累加器，不管在 ZF 中的数据是否是 00H，通过使用零测试函数，传送后比较指令〔CMP A, 00H〕可以省略，程序的步数可以减少。

```
LD      A, (P0)         ; A ← Port P0
JR      Z, SNEXT        ; If A == 00H then jump to SNEXT
```

3.1.2 LD (HL + d), n

本指令用于将立即数存入指定的存储空间的地址中。

例：将数据存放到地址为 00FCH 到 0102H 的 RAM 中。

```
LD      HL,   00FCH
LD      (HL + 0x01),    11110000B
LD      (HL + 0x01),    11011000B
LD      (HL + 0x01),    11111111B
LD      (HL + 0x01),    00001111B
LD      (HL + 0x01),    10100110B
LD      (HL + 0x01),    00110001B
LD      (HL + 0x01),    11100100B
```

注：当内存地址为 0000H 到 00FFH 时使用指令〔LD (x), n〕。

例：I/O 口初始化

```
LD      (00H), 11111111B        ;Port P0 ← 11111111B
LD      (01H), 11111111B        ;Port P1 ← 11111111B
LD      (02H), 00000111B        ;Port P2 ← 00000111B
LD      (03H), 11111111B        ;Port P3 ← 11111111B
```

3.1.3 XCH r, (HL + d)

本指令用于字节数据的位移。

例：将累加器 A 的内容移到地址为 0120H 到 0127H 的 RAM 中。

```
        LD      HL,  0120H
        LD      C,   07H
SSHIFT: XCH     A,   (HL + 1)
        DEC     C
        JRS     F,   SSHIFT
```

3.2 算术和逻辑运算指令

3.2.1 CMP (x), n

本指令用来比较内存数据。

例：审核 RAM 地址为 96H 是否为 0001010B。

```
        CMP     (96H), 00001010B
        JRS     F, SNE
SEQ: .....                      ; 相等
     :
SNE: .....                      ; 不相等
```

3.2.2 DAA r 和 DAS r

DAA 指令用于加 BCD 数据时对十进制制的调整。DAS 指令用于减 BCD 数据时对十进制制的调整。

例：存放在 WA 寄存器中对 4 位 BCD 数据加法。

```
ADD     A,   1          ; A ← A + 1
DAA     A
ADDC    W,   0          ; W ← W + CF
DAA     W
```

3.2.3 MUL H, L

DAA 本指令将 8 位寄存器 H 和 L 的内容相乘得到 16 位数的积。在多位数乘法的情况下，如同常规乘法那样计算。

例：将存放在 RAM 地址为 0072H-0073H 中的 16 位数据和存放在 WA 寄存器中的 16 位数据相乘，得到一个 32 位数的结果（积的低 16 位存放在 WA 寄存器中，高 16 位存放在 BC 寄存器中）。

```
LD      BC,  (72H)      ; BC ← RAM(0073H, 0072H)
LD      DE,  BC         ; HL ← C×A, BC ← B×W
LD      HL,  WA
XCH     H,   C
MUL     H,   L
MUL     B,   C
XCH     A,   E          ; WA ← W×C, DE ← B×A
MUL     W,   A
MUL     D,   E
ADD     WA,  DE         ; WA ← WA + DE
ADDC    B,   00H        ; B ← B + CF
LD      D,   C          ; WA ← WA + CH
LD      E,   H
ADD     WA,  DE
ADDC    B,   00H        ; B ← B + CF
LD      C,   W          ; C ← W
LD      W,   A          ; W ← A
LD      A,   L          ; A ← L
```

```
                          W       A
×)________________________B       C
                          C   ×   A
                      W   ×   C
                      B   ×   A
×)____________B   ×   W
              B       C       W       A
```

3.2.4 DIV WA, C

本指令将 2 字节的数据除以 1 字节的数据，并得到 1 字节的商和余数；当商超过 1 字节时产生溢出错误。所以除法分二步进行（见例 1 ）。当多字节数据除以 1 字节数据时，求商从最高位开始，与手工计算次序相同（见例 2)。DIV 指令对执行转换很方便，如将二进制转换成二-十进制（见例 3)。

例 1 ：计算 16 位数据（WA 寄存器）÷8 位数据（C 寄存器），并得到 16 位的商（WA 寄存器）和 8 位余数（B 寄存器)。

```
        LD      B,    00H       ; 如 C=0, 则跳至 SERROR
        CMP     C,    B
        JR      T,    SERROR
        XCH     A,    W
        XCH     W,    B
        DIV     WA,   C
        XCH     A,    B
        DIV     WA,   C         ; A ← 商的低字节
        XCH     W,    B         ; W ← 商的高字节
         :
SERROR: ......                  ; 零除出错
```

例 2:除存放在从 C 寄存器的内容指向的 HL 寄存器的内容所指定地址开始的 3 字节内存中的被除数，将商存放在紧跟在被除术后的 3 字节的内存中，并将余数存放在 W 寄存器中。

```
        LD      A,    (HL + 1)      ; A ← 被除数最高位数
        LD      W,    00H
        DIV     WA,   C             ; A ← 商的最高位数
        LD      (HL + 2),   A
        LD      A,    (HL + 1)
        DIV     WA,   C
        LD      (HL + 2),   A
        LD      A,    (HL)          ; A ← 被除数最低位数
        DIV     WA,   C             ; A ← 商的最低位数, W ← 余数
        LD      (HL + 3), A
```

例 3：将存储在 A 寄存器中的二进制术转换成 BCD 二-十进制，并将结果存放在 WA 寄存器。

```
LD      BC,  000AH      ; B ← 0, C ← 10
LD      W,   B          ; B ← 最低位数(1)
DIV     WA,  C
XCH     W,   B
DIV     WA,  C          ; A ← 中间位数(10)
XCH     A,   W          ; W ← 最高位数(100)
SWAP    A
ADD     A,   B          ; A ← 中间位数(10)和最低位数(1)
```

3.3 移位、循环和半字节操作指令

3.3.1 ROLD A, (HL + 1)

本指令适用于 BCD 二-十进制数据的移位。

例：将存放在累加器低半字节中的 BCD 数据移到地址为 00C8H-00CFH 的 RAM 中。

```
        LD      HL,  00C8H
        LD      C,   07H
SSHIFT: ROLD    A,   (HL + 1)
        DEC     C
        FRS     F,   SSHIFT
```

3.4 位和标志操作类指令

3.4.1 CLR (x).b

位设置/清除/求补指令有位测试功能，例如〔CLR (x).b〕装载内存地址 x 的 b 所指定的位的值的补码到 ZF 和 JF，然后再将所指定的位清为 0。

例：将内存地址 E9H 的第 3 位清为 0。如果其设为 1，则调用子程序。

```
        CLR     (0E9H).3
        JRS     T,   SSKIP
        CALL    PSUB
SSKIP:  ......
```

3.4.2 TEST (HL).A

本指令用于位搜寻。

例：从最高位开始搜索内存地址 0135H 的内容，将置为 1 的位的位置存放在累加器中。当没有 1 的位时，累加器置为 FFH。

```
        LD      HL,  0135H          ; HL ← memory address
        LD      A.7                 ; A ← bit number
SSEARCH: TEST   (HL). A
        JRS     F,   SEND
        DEC     A
        JRS     F,   SSEARCH
SEND:   ......
```

3.5 跳转指令

3.5.1 JRS T, a

当在传送/交换这类将 JF 置成 1 的指令之后，立即执行无条件跳转。跳转目的地在跳转指令所在地址的-14 到+17 范围之内。

例：无条件跳转至 SLABLE

```
        LD      A,   5CH
        JRS     T,   SLABLE
         :
SLABLE:  ......
```

3.5.2 JP (PC+A) 和 CALL (PC+A)

这些指令用于根据累加器的内容做多向分歧。

例：根据不同的累加器内容跳转至以下所示的地方，否则跳转至 SLABEL0。

当 01H 时，跳转至 SLABEL1
当 02H 时，跳转至 SLABEL2
当 03H 时，跳转至 SLABEL3
当 04H 时，跳转至 SLABEL4

```
        CMP     A,    05H
        JR      LT,   SKIP
        LD      A,    00H
SKIP:   SHLC    A
        JR      (PC + A)
        DW      SLABEL0, SLABEL1, SLABEL2, SLABEL3, SLABEL4
```

3.5.3 CALLV n

在子程序有 3 个或以上的地方使用 CALLV 指令转子可以提高内存使用率。

例：使用 CALLV 指令

```
        CSEG    REL,  CSAMPLE
        :
        :
        CALLV   PSUB1
        :
        :
        CALLV   PSUB2
        LD      A,    00H
PSUB1:  RET
PSUB2:  RET

        CSEG    ABS,  VECTORTABLE (0FFC0H)
PSUB1:  DW      VSUB1
PSUB2:  DW      VSUB2

        END
```

4. 指令集清單

i87 系列单片机具有 133 种类型的数据操作指令，提供 732 条指令。完整的 i87 指令集清單，共分成以下五大類：

(1) 数据传送、交换类指令
(2) 算术和逻辑运算类指令
(3) 移位、循环和半字节操作类指令
(4) 位和标志操作类指令
(5) 转移位操作类指令

詳細清單，請參閱「i87 Series Instruction Set」文件。

4.1 数据传送、交换类指令

有關数据传送、交换类指令之詳細指令清單，請參閱「i87 Series Instruction Set」文件之第 19 頁至第 25 頁。(2.1 Load/Store and Exchange Instructions)

4.2 算术和逻辑运算类指令

有關算术和逻辑运算类指令之詳細指令清單，請參閱「i87 Series Instruction Set」文件之第 27 頁至第 49 頁。(2.2 ALU Instructions)

4.3 移位、循环和半字节操作类指令

有關移位、循环和半字节操作类指令之詳細指令清單，請參閱「i87 Series Instruction Set」文件之第 51 頁至第 54 頁。(2.3 Shift, Rotate and Nibble Manipulation Instructions)

4.4 位和标志操作类指令

有關位和标志操作类指令之詳細指令清單，請參閱「i87 Series Instruction Set」文件之第 55 頁至第 65 頁。(2.4 Bit and Flag Manipulation Instructions)

4.5 转移位操作类指令

有關转移位操作类指令之詳細指令清單，請參閱「i87 Series Instruction Set」文件之第 67 頁至第 69 頁。(2.5 Jump Instructions)