M9 指令集

助记符 操作数	说明	周期数	受影响的状态位	注
字节操作类指令				
ADDAR R, d, a	A 与 R 相加	1	C, DC, Z, OV, N	1, 2
ADCAR R, d, a	A 带进位与 R 相加	1	C, DC, Z, OV, N	1, 2
ANDAR R, d, a	A 和 R 作逻辑与运算	1	Z, N	1, 2
CLRR R, a	将 R 清零	1	Z	2
COMR R, d, a	对 R 取反	1	Z, N	1, 2
JERA R, a	R 与 A 比较，相等跳过	1(2, 3)	无	3
JGRA R, a	R 与 A 比较，大于跳过	1(2, 3)	无	3
JLRA R, a	R 与 A 比较，小于跳过	1(2, 3)	无	1
DECR R, d, a	R 递减 1	1	C, DC, Z, OV, N	1, 2, 3
DJZR R, d, a	R 递减 1，为 0 跳过	1(2, 3)	无	1, 2, 3
DJNZR R, d, a	R 递减 1，非 0 跳过	1(2, 3)	无	1
INCR R, d, a	R 递增 1	1	C, DC, Z, OV, N	1, 2, 3
JZR R, d, a	R 递增 1，为 0 跳过	1(2, 3)	无	3
JNZR R, d, a	R 递增 1，非 0 跳过	1(2, 3)	无	1
ORAR R, d, a	A 与 R 作逻辑或运算	1	Z, N	1
MOVR R, d, a	传送 R	1	Z, N	1
MOVRR Rs,Rd	从源 Rs 送到目标 Rd	2	无	
MOVAR R,a	将 A 中的内容送入 R	1	无	
MULAR R,a	AREG 与 R 相乘	1	无	1
NEGR R,a	对 R 取补	1	C, DC, Z, OV, N	
RLR R, d, a	R 带进位循环左移	1	C, Z, N	1
RLNCR R, d, a	R 循环左移(无进位)	1	Z, N	
RRR R, d, a	R 带进位循环右移	1	C, Z, N	
RRNCR R, d, a	R 循环右移(无进位)	1	Z, N	
SETR R,a	将 R 全置为 1	1	无	1
SBCAR R, d, a	A 减去 R（带借位）	1	C, DC, Z, OV, N	
SUBRA R, d, a	R 减去 A	1	C, DC, Z, OV, N	1
SBCRA R, d, a	R 减去 A（带借位）	1	C, DC, Z, OV, N	
SWAPR R, d, a	对 R 进行半字节交换	1	无	3
JREZ R,a	测试 R，为 0 则跳过	1(2, 3)	无	1
XORAR R, d, a	A 和 R 作逻辑异或运算	1	Z, N	
位操作类指令				
BCLR R, b, a	将 R 中的指定位清零	1	无	1
BSET R, b, a	将 R 中的指定位置 1	1	无	1,
JBTS0 R, b, a	测试 R 的位，为 0 跳过	1(2, 3)	无	2, 3
JBTS1 R, b, a	测试 R 的位，为 1 跳过	1(2, 3)	无	2, 3
BNEG R, b, a	将 R 中的指定位取反	1	无	1
控制操作类指令				

RJBC	Label	如果有进位则跳转	1（2）	无	
RJBN	Label	如果为负则跳转	1（2）	无	
RJBNC	Label	如果没有进位则跳转	1（2）	无	
RJBNN	Label	如果不为负则跳转	1（2）	无	
RJBNOV	Label	如果未溢出则跳转	1（2）	无	
RJBNZ	Label	如果不为零则跳转	1（2）	无	
RJBOV	Label	如果溢出则跳转	1（2）	无	
RGOTO	Label	无条件跳转	2	无	
RJBZ	Label	如果为零则跳转	1（2）	无	
CALL	k,s	调用子程序	2	无	
CLRWDT	-	将看门狗定时器清零	1	TO, PD	
DAA	-	对 A 进行十进制调整	1	C	
GOTO	k	跳转到地址	2	无	
NOP	-	无操作	1	无	
NOP	-	无操作	1	无	3
SPOP	-	弹出堆栈栈顶 TOS	1	无	
SPUSH	-	压入堆栈栈顶 TOS	1	无	
RCALL	Label	相对调用	2	无	
SRESET		软件器件复位	1	所有	
RETIE	s	中断返回并允许中断	2	GIE/GIEH,	
RETIA	k	返回并将 k 送入 A	2	无	
RETURN	s	从子程序返回	2	无	
立即数操作类指令					
ADDIA	k	A 与立即数相加	1	C, DC, Z, OV, N	
ANDIA	k	k 和 A 作逻辑与运算	1	Z, N	
ORIA	k	k 和 A 作逻辑或运算	1	Z, N	
LDFSR	R,k	传送立即数 k 到 FSR	2	无	
BANKBSR	k	将立即数送入 BSR	1	无	
MOVIA	k	将立即数送入 A	1	无	
MULIA	k	k 与 A 相乘	1	无	
SUBIA	k	k 减去 A	1	C, DC, Z, OV, N	
XORIA	k	k 与 A 作逻辑异或运算	1	Z, N	
存储器操作					
RDT*		表读	2	无	
RDT*+		表读，后递增		无	
RDT*-		表读，后递减		无	
RDT+*		表读，预递增		无	
WDT*		表写	2	无	

注

1： 如果程序计数器（PC）被修改或者条件测试为真，则该指令需要两个周期。第二个周期执行一条 NOP 指令。

2： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 NOP 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

字段　说明

a　快速操作 RAM 位

a = 0：快速操作 RAM 内的 存储单元（ BSR 寄存器被忽略）

a = 1：由 BSR 寄存器指定的存储区

（编译的时候 IDE 会根据寄存器地址自动选择 a 的值）

bbb　8 位文件寄存器内的位地址（ 0 至 7）。

BSR　存储区选择寄存器。用于选择当前的 RAM 存储区。

C、DC、Z、OV 和 N　ALU 状态位：进位、半进位、全零、溢出和负标志

d　目标寄存器选择位

d = 0：结果保存至 AREG 寄存器

d = 1：结果保存至寄存器 R

dest 目标寄存器：可以是 AREG 寄存器或指定的文件寄存器地址。

R　8 位寄存器地址（ 00h 至 FFh）。

Rs　12 位文件寄存器地址（ 000h 至 FFFh），源地址。

Rd　12 位文件寄存器地址（ 000h 至 FFFh），目标地址。

GIE　全局中断允许位(在芯片中不存在此位，相当于 GIEH+GIEL)。

k　立即数、常数或者标号（可能是 8 位、 12 位或 20 位的值）。

label　标号名称

here　标号名称

n　相对跳转指令的相对地址（二进制补码形式），或 CALL/跳转和 RETURN 指令的直接地址。

PC　程序计数器。

PCL　程序计数器低字节。

PCLATH　程序计数器高字节锁存器。

TO　超时位。

PD　掉电位。

PRODH　乘积结果的高字节。

PRODL　乘积结果的低字节。

s　快速调用 /返回模式选择位 s = 0：不对影子寄存器进行更新，也不用影子寄存器的内容更新其他寄存器 ,s = 1：将寄存器的值装入影子寄存器或将影子寄存器中的值装入寄存器（快速模式）

TBLPTR　表指针（指向程序存储器地址）。

TABLAT　8 位表锁存器。

TOS　栈顶。

u　未使用或未改变。

WDT　看门狗定时器。

AREG　工作寄存器（累加器）。

x　无关位（ 0 或 1）。汇编器将生成 x = 0 的代码。

(text)　text 的内容。

[expr]<n>　表示由指针 expr 指向的寄存器中的 bit n。

→　赋值。

< >　寄存器位域。

∈　表示属于某个集合。

ADDAR A 与R 相加
语法： ADDAR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (A) + (R) → dest
受影响的状态位： N、OV、C、DC 和Z
说明： 将A 的内容与R 寄存器的内容相加。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： ADDAR REG, 0, 0
执行指令前
A = 25h
REG = 42h
执行指令后
A = 67h
REG = 42h

ADCAR A 与R 带进位相加
语法： ADCAR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (A) + (R) + (C) → dest
受影响的状态位： N、OV、C、DC 和Z
说明： 将A 的内容、进位标志位与数据存储单元R 的内容相加。如果d 为0，结果存储在A 中。如果d 为1，结果储在数据存储单元R 中。
指令字数： 1
指令周期数： 1
示例： ADCAR REG, 0, 1
执行指令前
进位标志位= 1
REG = 08h
A = 48h
执行指令后
进位标志位= 0
REG = 08h
A = 50h

ANDAR 将A 和R 作逻辑与运算

语法： ANDAR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (A) & (R) → dest
受影响的状态位： N 和 Z
说明： 将A 的内容与寄存器R 的内容进行逻辑与运算。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R（默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： ANDAR REG, 0, 0
执行指令前
A = 25h
REG = 41h
执行指令后
A = 01h
REG = 41h

CLRR 将 R 清零
语法： CLRR R ,a
操作数： 0 ≤ R ≤ 255
a ∈ [0,1]
操作： 000h → R
1 → Z
受影响的状态位： Z
说明： 清零指定寄存器的内容。如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。
指令字数： 1
指令周期数： 1
示例： CLRR FLAG_REG, 1
执行指令前
FLAG_REG = 5Ah
执行指令后
FLAG_REG = 00h

COMR 对R 取反
语法： COMR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈[0,1]
a ∈ [0,1]
操作： (R) → dest

受影响的状态位： N 和Z
说明： 将寄存器R 的内容取反。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： COMR REG, 0, 0
执行指令前
REG = 15h
执行指令后
REG = 15h
A = EAh

JERA 比较R 和A，如果R = A 则跳过
语法： JERA R ,a
操作数： 0 ≤R ≤255
a ∈ [0,1]
操作： (R) – (A)，
如果(R) = (A)，则跳过（无符号比较）
受影响的状态位： 无
说明： 通过执行无符号的减法，将数据存储单元R 的内容与A 的内容作比较。如果R = A，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2字指令，则执行JERA 需要3 个周期。
示例： HERE JERA REG, 0
NEQUAL :
EQUAL :
执行指令前
PC 地址= HERE
A = ？
REG = ？
执行指令后
如果REG = A；
PC = 地址（EQUAL）
如果REG ≠ A；
PC = 地址（NEQUAL）

JGRA 比较R 和A，如果R > A 则跳过
语法： JGRA R ,a

操作数： 0 ≤R ≤255
a ∈ [0,1]
操作： (R) – (A)，
如果(R) > (A)，则跳过（无符号比较）
受影响的状态位： 无
说明： 通过执行无符号的减法，将数据存储单元R 的内容与A 的内容作比较。如果R > A，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2 字指令，则执行JGRA 需要3 个周期。
示例： HERE JGRA REG, 0
NGREATER :
GREATER:
执行指令前
PC = 地址（HERE）
A = ？
执行指令后
如果REG A；
PC = 地址（GREATER）
如果REG ≤ A；
PC = 地址（NGREATER）

JLRA 比较R 和A，如果R < A 则跳过
语法： JLRA R, a
操作数： 0 ≤ R ≤ 255
a ∈ [0,1]
操作： (R) – （A）
如果(R) < (A)，则跳过（无符号比较）
受影响的状态位： 无
说明： 通过执行无符号的减法，将数据存储单元R 的内容与A 的内容作比较。如果R < A，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2字指令，
则执行JLRA 需要3 个周期。
示例： HERE JLRA REG, 1
NLESS:
LESS:
执行指令前

PC = 地址（HERE）
A = ？
执行指令后
如果REG < A ；
PC = 地址（LESS）
如果REG ≥ A；
PC = 地址（NLESS）

DECR R 递减1
语法： DECR R, d, a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R) – 1 → dest
受影响的状态位： C、DC、N、OV 和Z
说明： 将寄存器R 的内容递减1。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： DECR CNT, 1, 0
执行指令前
CNT = 01h
Z = 0
执行指令后
CNT = 00h
Z = 1

DJZR R 递减1，为0 则跳过
语法： DJZR R, d, a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R) – 1 → dest，
如果结果 = 0 则跳过
受影响的状态位： 无
说明： 将寄存器R 的内容递减1。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果结果为0，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）

注： 如果跳过，且后面跟有2字指令，
则执行DJZR 需要3 个周期。
示例： HERE DJZR CNT, 1, 1
GOTO LOOP
CONTINUE
执行指令前
PC = 地址（HERE）
执行指令后
CNT = CNT - 1
如果CNT = 0 ；
PC = 地址（CONTINUE）
如果CNT ≠ 0 ；
PC = 地址（HERE + 2）

DJNZR R 递减1，非0 则跳过
语法： DJNZR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R) – 1→ dest，
如果结果 ≠ 0 则跳过
受影响的状态位： 无
说明： 将寄存器R 的内容递减1。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果结果不为0，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2字指令，则执行DJNZR 需要3 个周期。
示例： HERE DJNZR TEMP, 1, 0
ZERO :
NZERO :
执行指令前
TEMP = ？
执行指令后
TEMP = TEMP – 1，
如果TEMP = 0 ；
PC = 地址（ZERO）
如果TEMP ≠ 0 ；
PC = 地址（NZERO）

INCR R递增1

语法： INCR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R) + 1 → dest
受影响的状态位： C、DC、N、OV 和Z
说明： 将寄存器R 的内容递增1。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： INCR CNT, 1, 0
执行指令前
CNT = FFh
Z = 0
C = ？
DC = ？
执行指令后
CNT = 00h
Z = 1
C = 1
DC = 1

JZR R递增1，为0 则跳过
语法： JZR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R) + 1 → dest，
如果结果 = 0 则跳过
受影响的状态位： 无
说明： 将寄存器R 的内容递增1。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果结果为0，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2 字指令，则执行JZR 需要3 个周期。
示例： HERE JZR CNT, 1, 0
NZERO :
ZERO :
执行指令前
PC = 地址（HERE）

执行指令后
CNT = CNT + 1
如果CNT = 0 ；
PC = 地址（ZERO）
如果CNT ≠ 0 ；
PC = 地址（NZERO）

JNZR R 递增1，非0 则跳过
语法： JNZR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈[0,1]
a ∈ [0,1]
操作： (R) + 1 → dest，
如果结果 ≠ 0 则跳过
受影响的状态位： 无
说明： 将寄存器R 的内容递增1。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果结果不为0，则丢弃已取的指令转而执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2字指令，则执行JNZR 需要3 个周期。
示例： HERE JNZR REG, 1, 0
ZERO
NZERO
执行指令前
PC = 地址（HERE）
执行指令后
REG = REG + 1
如果REG≠ 0 ；
PC = 地址（NZERO）
如果REG= 0 ；
PC = 地址（ZERO）

ORAR 将A 与R 作逻辑或运算
语法： ORAR R ,d ,a
操作数： 0 ≤ R ≤ 255
d∈ [0,1]
a∈ [0,1]
操作： (A) | (R) → dest
受影响的状态位： N 和Z
说明： 将A 的内容与寄存器R 的内容进行逻辑或运算。如果d 为0，结果存储

在A 中。如果d 为1，结果存回寄存器R（默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： ORAR RESULT, 0, 1
执行指令前
RESULT = 13h
A = 91h
执行指令后
RESULT = 13h
A = 93h

MOVR 传送R
语法： MOVR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： R → dest
受影响的状态位： N 和 Z
说明： 根据d 的状态，将寄存器R 的内容送入目标寄存器。如果d 为0，结果存储在A中。如果d 为1，结果存回寄存器R （默认）。R 可以为256 字节存储区中的任何地址单元。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： MOVR REG, 0, 0
执行指令前
REG = 15h
A = F5h
执行指令后
REG = 15h
A = 15h

MOVRR 将源寄存器的内容送入目标寄存器
语法： MOVRR Rs,Rd
操作数： 0 ≤ R ≤ 4096
0≤ Rd ≤ 4095
操作： (Rs) → Rd
受影响的状态位： 无
说明： 将源寄存器Rs 的内容送入目标寄存器Rd。源寄存器Rs 可以是4096 字节数据空间（000h 至FFFh）中的任何单元，目标寄存器Rd 也可以是000h 至FFFh 中的任何单元。源或目标寄存器都可以是A。MOVRR 指令对于将数据存

储单元中的内容送入外设寄存器（如发送缓冲区或I/O端口）的场合非常有用。
MOVRR 指令不能使用PCL、TOSU、TOSH 或TOSL 作为目标寄存器。
指令字数： 2
指令周期数： 2 （3）
示例： MOVRR REG1, REG2
执行指令前
REG1 = 33h
REG2 = 11h
执行指令后
REG1 = 33h
REG2 = 33h

MOVAR 将A 的内容送入R
语法： MOVAR R ,a
操作数： 0 ≤R≤255
a ∈ [0,1]
操作： (A) → R
受影响的状态位： 无
说明： 将A 寄存器中的数据送入寄存器R。R 可以是256字节存储区中的任何地址单元。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： MOVAR REG, 0
执行指令前
A = 44h
REG = 00h
执行指令后
A = 44h
REG = 44h

MULAR 将 A 与 R 的内容相乘
语法： MULAR R ,a
操作数： $0 \le R \le 255$
a ∈ [0,1]
操作： (A) x (R) → PRODH:PRODL
受影响的状态位： 无
说明： 将 A 的内容与寄存器单元 R 的内容执行无符号的乘法运算。运算的 16 位结果保存在 PRODH:PRODL 寄存器对中，其中 PRODH 用于存储高字节。A 和 R 的内容都不改变。所有状态标志位都不受影响。请注意此操作不可能发生溢出或进位。结果有可能为全零，但不会被检测到。如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

指令字数： 1
指令周期数： 1
示例： MULAR REG, 1
执行指令前
A = C8h
REG = A5h
PRODH = ？
PRODL = ？
执行指令后
A = C8h
REG = A5h
PRODH = 80h
PRODL = E8h

NEGR 对 R 取补
语法： NEGR R,a
操作数： $0 \leq R \leq 255$
a ∈ [0,1]
操作： (R) + 1 → R
受影响的状态位： N、OV、C、DC 和 Z
说明： 用二进制补码对存储单元 R 取补，结果存储在数据存储单元 R 中。如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。
指令字数： 1
指令周期数： 1
示例： NEGR REG, 1
执行指令前
REG = 0101 1010 [5Ah]
执行指令后
REG = 10100 0110 [A6h]

RLR R 带进位循环左移
语法： RLR R ,d ,a
操作数： 0 ≤R ≤255
d ∈ [0,1]
a ∈ [0,1]
操作： (R<n>) → dest<n + 1>，
(R<7>) → C，
(C) → dest<0>
受影响的状态位： C、N 和 Z
说明： 将寄存器R的内容连同进位标志位一起循环左移1 位。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R（默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。

指令字数： 1
指令周期数： 1
示例： RLR REG, 0, 0
执行指令前
REG = 1100 0011
C = 0
执行指令后
REG = 1100 0011
A = 1000 0110
C = 1

RLNCR R 循环左移（不带进位）
语法： RLNCR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R<n>) → dest<n + 1>，
(R<7>) → dest<0>
受影响的状态位： N 和 Z
说明： 将寄存器 R 的内容循环左移 1 位。如果 d 为 0，结果存储在 A 中。如果 d 为 1，结果存回寄存器 R （默认）。如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。
指令字数： 1
指令周期数： 1
示例： RLNCR REG, 1, 0
执行指令前
REG = 1010 1010
执行指令后
REG = 0101 0101
寄存器 R

RRR R 带进位循环右移
语法： RRR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R<n>) → dest<n – 1>，
(R<0>) → C，
(C) → dest<7>
受影响的状态位： C、N 和 Z
说明： 将寄存器 R 的内容连同进位标志位一起循环右移 1 位。如果 d 为 0，结果存储在 A 中。如果 d 为 1，结果存回寄存器 R（默认）。如果 a 为 0，选择快

速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。
指令字数： 1
指令周期数： 1
示例： RRR REG, 0, 0
执行指令前
REG = 1010 0101
C = 0
执行指令后
REG = 1010 0101
A = 0101 0010
C = 1

RRNCR R 循环右移（不带进位）
语法： RRNCR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R<n>) → dest<n – 1>，
(R<0>)→ dest<7>
受影响的状态位： N 和 Z
说明： 将寄存器R 的内容循环右移1 位。如果d为0，结果存储在A中。如果d为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区（默认），忽略BSR 的值。如果a 为1，则根据BSR 值选择存储区。
指令字数： 1
指令周期数： 1
例1： RRNCR REG, 1, 0
执行指令前
REG = 1001 0110
执行指令后
REG = 0100 1011

SETR 将R 的内容置为全1
语法： SETR R,a
操作数： 0 ≤ R ≤ 255
a ∈ [0,1]
操作： FFh → R
受影响的状态位： 无
说明： 将指定寄存器的内容置为FFh。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： SETR REG, 1

执行指令前
REG = 55h
执行指令后
REG = FFh

SBCAR A 减去R （带借位）
语法： SBCAR R ,d ,a
操作数： 0≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (A) – (R) – ($\overline{C}$) → dest

受影响的状态位： N、OV、C、DC 和Z
说明： 将A的内容减去R 寄存器的内容和进位标志位（借位）（通过二进制补码方式
进行运算）。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR选择GPR存储区。
指令字数： 1
指令周期数： 1
例： SBCAR REG, 1, 0
执行指令前
REG = 3
A = 2
C = 1
执行指令后
REG = FF
A = 2
C = 0
Z = 0
N = 1 ；结果为负

SUBRA R 减去A
语法： SUBRA R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈ [0,1]
a ∈ [0,1]
操作： (R) – (A) → dest
受影响的状态位： N、OV、C、DC 和 Z
说明： 用寄存器R 中的内容减去A寄存器的内容（通过二进制补码方式进行运算）。如果d 为0，结果存储在A 中。如果d为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR选择GPR存储

区。
指令字数： 1
指令周期数： 1
例： SUBRA REG, 1, 0
执行指令前
REG = 3
A = 2
C = ？
执行指令后
REG = 1
A = 2
C = 1 ；结果为正
Z = 0
N = 0

SBCRA R 减去A （带借位）
语法： SBCRA R ,d ,a
操作数： 0 ≤R ≤255
d ∈[0,1]
a ∈ [0,1]

操作： $(R) - (A) - (\overline{C}) \rightarrow$ dest

受影响的状态位： N、OV、C、DC 和Z
说明： 用R 寄存器的内容减去A的内容和进位标志位（借位）（通过二进制补码方式进行运算）。如果d 为0，结果存储在A 中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
例： SBCRA REG, 1, 0
执行指令前
REG = 19h （0001 1001）
A = 0Dh （0000 1101）
C = 1
执行指令后
REG = 0Ch （0000 1100）
A = 0Dh （0000 1101）
C = 1
Z = 0
N = 0 ；结果为正

SWAPR 将R 的高半字节和低半字节交换

语法： SWAPR R ,d ,a
操作数： 0 ≤ R ≤ 255
d ∈[0,1]
a ∈ [0,1]
操作： (R<3:0>) → dest<7:4>，
(R<7:4>) → dest<3:0>
受影响的状态位： 无
说明： 寄存器R 的高半字节和低半字节相互交换。如果d 为0，结果存储在A中。如果d 为1，结果存回寄存器R （默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： SWAPR REG, 1, 0
执行指令前
REG = C3h
执行指令后
REG = 3Ch

JREZ 测试R，为0 则跳过
语法： JREZ R ,a
操作数： 0 ≤ R ≤ 255
a ∈ [0,1]
操作： R = 0 则跳过
受影响的状态位： 无
说明： 如果R = 0，丢弃执行当前指令过程中已取的下一条指令并执行一条NOP 指令，使该指令成为双周期指令。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2字指令，则执行JREZ 需要3 个周期。
示例： HERE JREZ CNT, 1
NZERO :
ZERO :
执行指令前
PC = 地址（HERE）
执行指令后
如果CNT = 00h，
PC = 地址（ZERO）
如果CNT ≠ 00h，
PC = 地址（NZERO）

XORAR 将A 与R 作逻辑异或运算

语法： XORAR R ,d ,a
操作数：0 ≤R ≤255
d ∈[0,1]
a ∈ [0,1]
操作： (A) ^(R) → dest
受影响的状态位： N 和 Z
说明： 将A的内容与寄存器R 的内容进行逻辑异或运算。如果d 为0，结果存储在A中。如果d 为1，结果存回寄存器R（默认）。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： XORAR REG, 1, 0
执行指令前
REG = AFh
A = B5h
执行指令后
REG = 1Ah
A = B5h

BCLR 将R 中的某位清0
语法： BCLR R, b ,a
操作数： 0 ≤ R ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]
操作： 0 → R<b>
受影响的状态位： 无
说明： 将寄存器R 的位b 清0。
如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： BSET FLAG_REG, 7, 1
执行指令前
FLAG_REG = 82h
执行指令后
FLAG_REG = 02h

BSET 将R 中的某位置1
语法： BSET R, b ,a
操作数： 0 ≤ R ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]
操作： 1 → R<b>

受影响的状态位： 无
说明： 将寄存器R 的位b 置1。
如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： BSET FLAG_REG, 7, 1
执行指令前
FLAG_REG = 02h
执行指令后
FLAG_REG = 82h

JBTS0 测试文件寄存器中的某位，为0则跳过
语法： JBTS0 R, b ,a
操作数： 0 ≤ R ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]
操作： 如果(R<b>) = 0，则跳过
受影响的状态位： 无
说明： 如果寄存器R 的位b 为0，则跳过下一条指令。即在b 位为0 时，丢弃执行当前指令过程中已取的下一条指令，转而执行一条NOP 指令，使该指令成为双周期指令。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2 字指令，
则执行JBTS0 需要3 个周期。
示例： HERE
FALSE
TRUE
JBTS0 FLAG, 1, 0
执行指令前
PC = 地址（HERE）
执行指令后
如果FLAG<1> = 0;
PC = 地址（TRUE）
如果FLAG<1> = 1;
PC = 地址（FALSE）

JBTS1 测试文件寄存器中的某位，为1则跳过
语法： JBTS1 R, b ,a
操作数： 0 ≤ R ≤ 255
0 ≤b ≤ 7
a ∈ [0,1]

操作： 如果(R<b>) = 1，则跳过
受影响的状态位： 无
说明： 如果寄存器R 的位b 为1，则跳过下一条指令。即在b 位为1 时，丢弃执行当前指令过程中已取的下一条指令，转而执行一条NOP 指令，使该指令成为双周期指令。
如果a 为0，选择快速操作存储区。如果a为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1 （2）
注： 如果跳过，且后面跟有2 字指令，
则执行JBTS1 需要3 个周期。
示例： HERE
FALSE
TRUE
JBTS1 FLAG, 1, 0
执行指令前
PC = 地址（HERE）
执行指令后
如果FLAG<1> = 0;
PC = 地址（FALSE）
如果FLAG<1> = 1;
PC = 地址（TRUE）

BNEG 将R 中的某位取反
语法： BNEG R, b ,a
操作数： 0 ≤ R ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]
操作： (~R<b>) → R<b>
受影响的状态位： 无
说明： 将数据存储单元R 中的位b 取反。如果a 为0，选择快速操作存储区。如果a 为1，使用BSR 选择GPR 存储区。
指令字数： 1
指令周期数： 1
示例： BNEG PORTC, 4, 0
执行指令前：
PORTC = 0111 0101 [72h]
执行指令后：
PORTC = 0110 0101 [62h]

RJBC 进位则跳转
语法： RJBC label
操作： 如果进位标志位为1

PC（label） → PC
受影响的状态位： 无
说明： 如果进位标志位为1，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n”（以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n (-128 ≤ n ≤ 127) 。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBC Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果进位标志位= 1 ；
PC = 地址（Jump）
如果进位标志位= 0 ；
PC = 地址（HERE + 2）

RJBN 进位则跳转
语法： RJBN label
操作： 如果负标志位为1
PC（label） → PC
受影响的状态位： 无
说明： 如果负标志位为1，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n”（以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBN Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果进位标志位= 1 ；
PC = 地址（Jump）
如果进位标志位= 0 ；
PC = 地址（HERE + 2）

RJBNC 无进位则跳转
语法： RJBNC label
操作： 如果进位标志位为0
PC（label） → PC
受影响的状态位： 无

说明： 如果进位标志位为0，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n”（以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBNC Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果进位标志位= 0 ；
PC = 地址（Jump）
如果进位标志位= 1 ；
PC = 地址（HERE + 2）

RJBNN 不为负则跳转
语法： RJBNN label
操作： 如果负标志位为0
PC（label） → PC
受影响的状态位： 无
说明： 如果负标志位为0，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n”（以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBNN Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果负标志位= 0 ；
PC = 地址（Jump）
如果负标志位= 1 ；
PC = 地址（HERE + 2）

RJBNOV 不溢出则跳转
语法： RJBNOV label
操作： 如果溢出标志位为0
PC（label） → PC
受影响的状态位： 无
说明： 如果溢出标志位为0，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n”（以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条

指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBNOV Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果溢出标志位= 0 ；
PC = 地址（Jump）
如果溢出标志位= 1 ；
PC = 地址（HERE + 2）

RJBNZ 不为零则跳转
语法： RJBNZ label
操作： 如果全零标志位为0
PC（label） → PC
受影响的状态位： 无
说明： 如果全零标志位为0，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n” （以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBNZ Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果全零标志位= 0 ；
PC = 地址（Jump）
如果全零标志位= 1 ；
PC = 地址（HERE + 2）

RJBOV 溢出则跳转
语法： RJBOV label
操作： 如果溢出标志位为1
PC（label）→ PC
受影响的状态位： 无
说明： 如果溢出标志位为1，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n” （以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。

指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBOV Jump
执行指令前
PC = 地址（HERE）
执行指令后
如果溢出标志位= 1 ；
PC = 地址（Jump）
如果溢出标志位= 0 ；
PC = 地址（HERE + 2）

RGOTO 无条件跳转
语法： RGOTO label
操作：
PC（label） → PC
受影响的状态位： 无
说明：如果写作$+2n，那么跳转的指令为n条，“2n” （以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-1024 ≤ n ≤ 1023)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 2
示例： HERE RGOTO Jump
执行指令前
PC = 地址（HERE）
执行指令后
PC = 地址（Jump）

RJBZ 为零则跳转
语法： RJBZ label
操作： 如果全零标志位为1
PC（label） → PC
受影响的状态位： 无
说明： 如果全零标志位为1，程序将跳转。如果写作$+2n，那么跳转的指令为n条，“2n” （以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-128 ≤ n ≤ 127)。该指令为一条双周期指令。
指令字数： 1
指令周期数： 1 （2）
示例： HERE RJBZ Jump
执行指令前
PC = 地址（HERE）
执行指令后

如果全零标志位= 1 ；
PC = 地址（Jump）
如果全零标志位= 0 ；
PC = 地址（HERE + 2）

CALL 调用子程序
语法： CALL k ,s
操作数： 0 ≤ k ≤ 1048575
s ∈ [0,1]
操作： (PC) + 4 → TOS，
k → PC<20:1>，
如果s = 1
(A) → AS，
(STATUS) → STATUSS，
(BSR) → BSRS
受影响的状态位： 无
说明： 可在整个2 MB 的存储器范围内进行子程序调用。首先，将返回地址（PC + 4）压入返回堆栈。如果s = 1，还会将A、STATUS 和BSR 寄存器的内容存入它们各自的影子寄存器AS、STATUSS 和BSRS。如果s = 0，将不会进行任何更新（默认）。然后，将20 位的值k 装入PC<20:1>。CALL 是一条双周期指令。
指令字数： 2
指令周期数： 2
示例： HERE CALL THERE, 1
执行指令前
PC = 地址（HERE）
执行指令后
PC = 地址（THERE）
TOS = 地址（HERE + 4）
AS = A
BSRS = BSR
STATUSS= STATUS

CLRWDT 将看门狗定时器清零
语法： CLRWDT
操作数： 无
操作： 000h → WDT，
1 → TO，
1 → PD
受影响的状态位： TO 和 PD
说明： CLRWDT 指令复位看门狗定时器。状态位 TO 和 PD 置 1。
指令字数： 1

指令周期数： 1
示例： CLRWDT
执行指令前
WDT 计数器= ？
执行指令后
WDT 计数器= 00h
TO = 1
PD = 1

DAA 对A 寄存器进行十进制调整
语法： DAA
操作数： 无
操作： 如果[A<3:0> > 9] 或[DC = 1]，则
(A<3:0>) + 6 → A<3:0> ；
否则
(A<3:0>) → A<3:0> ；
如果[A<7:4> + DC > 9] 或[C = 1]，则
(A<7:4>) + 6 + DC →A<7:4> ；
否则
(A<7:4>) + DC →A<7:4>
受影响的状态位： C
说明： DAA 指令调整A 寄存器内的8 位值，即之前两个压缩BCD 格式的变量之和，并产生一个正确的压缩BCD 格式结果。
指令字数： 1
指令周期数： 1
例1：
DAA
执行指令前
A = A5h
C = 0
DC = 0
执行指令后
A = 05h
C = 1
DC = 0

GOTO 无条件跳转
语法： GOTO k
操作数： 0 ≤ k ≤ 1048575
操作： k → PC<20:1>
受影响的状态位： 无
说明： GOTO 指令允许无条件跳转到整个2 MB存储器范围中的任何位置。将

20 位值k 装入PC<20:1>。GOTO 始终为双周期指令。
指令字数： 2
指令周期数： 2
示例： GOTO THERE
执行指令后
PC = 地址（THERE）

NOP 空操作
语法： NOP
操作数： 无
操作： 无操作
受影响的状态位： 无
说明： 不执行任何操作。
指令字数： 1
指令周期数： 1
示例：NOP

SPOP 弹出返回堆栈栈顶的内容
语法： SPOP
操作数： 无
操作： (TOS) → 位桶（即丢弃）
受影响的状态位： 无
说明： 从返回堆栈弹出TOS 值并丢弃。然后，前一个压入返回堆栈的值成为TOS 值。此指令可以让用户正确管理返回堆栈，从而实现软件堆栈。
指令字数： 1
指令周期数： 1
示例： SPOP
GOTO NEW
执行指令前
TOS = 1234h
堆栈（下一级） = 4320h
执行指令后
TOS = 4320h
PC = NEW

SPUSH 将数据压入返回堆栈栈顶
语法： SPUSH
操作数： 无
操作： (PC + 2) → TOS
受影响的状态位： 无
说明： PC + 2 的值被压入返回堆栈的栈顶。原先的TOS 值被压入堆栈的下一

级。此指令允许通过修改TOS 并将其压入返回堆栈来实现软件堆栈。
指令字数： 1
指令周期数： 1
示例： SPUSH
执行指令前
TOS = 3456h
PC = 0122h
执行指令后
PC = 0124h
TOS = 0124h
堆栈（下一级） = 3456h

RCALL 相对调用
语法： RCALL Label
操作数：
操作： (PC) + 2 → TOS，
Label(PC) → PC
受影响的状态位： 无
说明： 从当前地址跳转（最多1K）来调用子程序。首先，将返回地址（PC + 2）压入返回堆栈。然后，将Label地址写入PC，Label也可以用$+2n代替，“2n”（以二进制补码表示）与PC 相加。由于PC 将递增以便取出下一条指令，所以新地址将为PC + 2 + 2n(-1024 ≤ n ≤ 1023) 。该指令为一条双周期指令。
指令字数： 1
指令周期数： 2
示例： HERE RCALL Jump
执行指令前
PC = 地址（HERE）
执行指令后
PC = 地址（Jump）
TOS = 地址（HERE + 2）

SRESET 复位
语法： SRESET
操作数： 无
操作： 将所有受MCLR 复位影响的寄存器和标志位复位。
受影响的状态位： 全部
说明： 此指令可实现用软件执行MCLR 复位。
指令字数： 1
指令周期数： 1
示例： SRESET
执行指令后
寄存器 = 复位值

标志位 = 复位值

RETIE 从中断返回
语法： RETIE s
操作数： s ∈[0,1]
操作： (TOS) → PC，
1 → GIEH 和GIEL，
如果s = 1
(AS) → A，
(STATUSS) → STATUS，
(BSRS) → BSR，
PCLATU 和PCLATH 保持不变
受影响的状态位： GIEH和GIEL
说明： 从中断返回。执行出栈操作，将栈顶（TOS）的内容装入PC。通过将高或低优先级全局中断允许位置1，来允许中断。如果s = 1，则影子寄存器AS、STATUSS和BSRS的内容将被装入对应的寄存器A、STATUS 和BSR。如果s= 0，则不更新这些寄存器（默认）。
指令字数： 1
指令周期数： 2
示例： RETIE 1
中断后
PC = TOS
A = AS
BSR = BSRS
STATUS = STATUSS
GIEH，GIEL = 1

RETIA 将立即数返回到A
语法： RETIA k
操作数： 0 ≤ k ≤ 255
操作： k → A，
(TOS) → PC，
PCLATU 和PCLATH 保持不变
受影响的状态位： 无
说明： 将8 位立即数k 装入A。将栈顶内容（返回地址）装入程序计数器。高字节地址锁存器（PCLATH）内容保持不变。
指令字数： 1
指令周期数： 2
示例：
RETIA kn
执行指令前
A = 07h

执行指令后
A = kn 的值

RETURN 从子程序返回
语法： RETURN s
操作数： s ∈ [0,1]
操作： (TOS) → PC，
如果s = 1
(AS) → A，
(STATUSS) →STATUS，
(BSRS) →BSR，
PCLATU 和PCLATH 保持不变
受影响的状态位： 无
说明： 从子程序返回。执行出栈操作，将栈顶（TOS）的内容装入程序计数器。如果s= 1，则影子寄存器AS、STATUSS 和BSRS的内容将被装入对应的寄存器A、STATUS 和BSR。如果s = 0，则不更新这些寄存器（默认）。
指令字数： 1
指令周期数： 2
示例： RETURN
执行指令后：
PC = TOS

ADDIA A 与立即数相加
语法： ADDIA k
操作数： 0 ≤ k ≤ 255
操作： (A) + k → A
受影响的状态位： N、OV、C、DC 和Z
说明： 将A寄存器的内容与8 位立即数k 相加，结果存储在A 寄存器中。
指令字数： 1
指令周期数： 1
示例： ADDIA 21h
执行指令前
A = 20h
执行指令后
A = 41h

ANDIA 立即数和A 寄存器作逻辑与运算
语法： ANDIA k
操作数：0 ≤ k ≤ 255
操作： (A) & k → A
受影响的状态位： N 和 Z

说明： 将A 的内容与8 位立即数k 进行逻辑与运算。结果存储在A 寄存器中。
指令字数： 1
指令周期数： 1
示例： ANDIA 05Ah
执行指令前
A = A5h
执行指令后
A = 00h

ORIA 将立即数与A 作逻辑或运算
语法： ORIA k
操作数： 0 ≤ R ≤ 255
操作： (A) | k → A
受影响的状态位： N 和 Z
说明： 将A 的内容与8 位立即数k 进行逻辑或运算。结果存储在A 寄存器中。
指令字数： 1
指令周期数： 1
示例： ORIA 35h
执行指令前
A = 9Ah
执行指令后
A = BFh

LDFSR 装入FSR
语法： LDFSR R, k
操作数： 0 ≤ R ≤ 2
0 ≤ k ≤ 4095
操作： k → FSR
受影响的状态位： 无
说明： 将12 位立即数k 装入R 所指向的FSR寄存器。
指令字数： 2
指令周期数： 2
示例： LDFSR 2, 3ABh
执行指令后
FSR2H = 03h
FSR2L = ABh

BANKBSR 将立即数送入BSR 的低半字节
语法： BANKBSR k

操作数： 0 ≤ k ≤ 15
操作： k → BSR
受影响的状态位： 无
说明： 将4 位立即数k 装入存储区选择寄存器（BSR）。
指令字数： 1
指令周期数： 1
示例： BANKBSR 5
执行指令前
BSR 寄存器= 02h
执行指令后
BSR 寄存器= 05h

MOVIA 将立即数送入A
语法： MOVIA k
操作数： 0 ≤ f ≤ 255
操作： k → A
受影响的状态位： 无
说明： 将8 位立即数k 装入A。
指令字数： 1
指令周期数： 1
示例： MOVIA 5Ah
执行指令后
A = 5Ah

MULIA 将立即数与 A 中的内容相乘
语法： MULIA k
操作数： $0 \le k \le 255$
操作： (A) * k → PRODH:PRODL
受影响的状态位： 无
说明： 将 A 的内容与 8 位立即数 k 进行无符号的乘法运算。16 位的结果存储在 PRODH:PRODL 寄存器对中，其中 PRODH 用于存储高字节。A 的内容不改变。所有状态标志位都不受影响。请注意此操作不可能发生溢出或进位。结果有可能为全零，但不会被检测到。
指令字数： 1
指令周期数： 1
示例： MULIA 0C9h
执行指令前
A = A2h
PRODH = ？
PRODL = ？
执行指令后
A = A2h
PRODH = 7Fh

PRODL = 32h

SUBIA 立即数减去A 的内容
语法： SUBIA k
操作数： 0 ≤ k ≤ 255
操作： k – (A) → A
受影响的状态位： N、OV、C、DC 和Z
说明：用8 位立即数k 减去A。结果存储在A寄存器中。
指令字数： 1
指令周期数： 1
例： SUBIA 02h
执行指令前
A = 01h
C = ？
执行指令后
A = 01h
C = 1 ；结果为正
Z = 0
N = 0

XORIA 将立即数与A 作逻辑异或运算
语法： XORIA k
操作数： 0 ≤ k ≤ 255
操作： (A)^k → A
受影响的状态位： N 和 Z
说明： 将A 的内容与8 位立即数k 进行逻辑异或运算。结果存储在A 寄存器中。
指令字数： 1
指令周期数： 1
示例： XORIA 0AFh
执行指令前
A = B5h
执行指令后
A = 1Ah

RDT 表读
语法： RDT (*; *+; *-; +*)
操作数： 无
操作： 如果执行RDT *，

(程序存储单元(TBLPTR)) → TABLAT ；
TBLPTR 不改变；
如果执行RDT *+，
(程序存储单元(TBLPTR)) → TABLAT ；
(TBLPTR) + 1 → TBLPTR ；
如果执行RDT *-，
(程序存储单元(TBLPTR)) →TABLAT ；
(TBLPTR) – 1 → TBLPTR ；
如果执行RDT +*，
(TBLPTR) + 1 → TBLPTR ；
(程序存储单元(TBLPTR)) →TABLAT ；
受影响的状态位：无
说明： 此指令用于读取程序存储单元（P.M.）的内容。使用表指针（TBLPTR）对程序存储单元进行寻址。TBLPTR指向程序存储器中的每个字节。TBLPTR[0] = 0： 程序存储字的最低有效字节TBLPTR[0] = 1： 程序存储字的最高有效字节
指令字数： 1
指令周期数： 2
例： RDT *+ ;
执行指令前
TABLAT = 55h
TBLPTR = 00A356h
存储单元（00A356h） = 34h
执行指令后
TABLAT = 34h
TBLPTR = 00A357h

WDT 表写
语法： WDT *
操作数： 无
操作： 表写
受影响的状态位： 无
说明： 执行表写指令后，开始烧录，详细步骤参见规格书表写章节。
指令字数： 1
指令周期数： 2