

QT8Sxxx

指令集说明

Version 1.05

2019年6月

本公司保留对产品可靠性、功能和设计方面的改进作进一步说明的权利。说明文档的更改，恕不另行通知。

修正记录

版本	日期	描述
Ver1.02	2014-5-15	补全指令说明
Ver1.03	2014-10-8	勘误
Ver1.04	2014-11-24	删除 ADCIA 指令
Ver1.05	2019-6-28	增加指令范例

目录

目录.....	3
1 QT8SXXX 指令集简述	5
1.1 概述	5
1.2 符号说明	5
2 QT8SXXX 指令集表	6
3 QT8SXXX 指令说明	8
ADDAR R,D.....	8
ADCAR R,D.....	9
SUBAR R,D.....	9
SBCAR R,D.....	10
SBCRA R,D.....	11
SUBRA R,D.....	10
ANDAR R,D.....	11
ORAR R,D.....	12
XORAR R,D.....	12
COMR R,D.....	13
MOVR R,D.....	13
MOVAR R.....	14
CLRR R.....	14
SWAPR R,D.....	15
RLR R,D.....	15
RLRNC R,D.....	16
RRR R,D.....	16
RRRNC R,D.....	17
DECR R,D.....	17
JNZR R,D.....	18
DJZR R,D.....	18
INCR R,D.....	19
JZR R,D.....	19
DJNZR R,D.....	20
JCMPPAR R.....	20
JNCMPAR R.....	21
JGAR R.....	21
JLAR R.....	22
XCHAR R.....	22
JBTS0 R,B.....	23
JBTS1 R,B.....	23
BCLR R,B.....	24
BSET R,B.....	24
ADDIA I.....	25

ADCIA	I	25
SUBIA	I	25
SBCIA	I	26
SUBAI	I	27
SBCAI	I	27
ANDIA	I	28
ORIA	I	28
XORIA	I	29
MOVIA	I	29
RETIA	I	30
JCMPAI	I	30
JNCMPAI	I	30
RLA		31
RLANC		32
RRA		32
RRANC		33
DECA		33
DJZA		34
INCA		34
JZA		35
RETIE		35
RETURN		36
NOP		36
RDT		37
DAA		38
DSA		38
PUSH		39
POP		39
CLRWDT		40
CALL	I	40
GOTO	I	41
ACALL	I	41
AGOTO	I	42

1 QT8Sxxx 指令集简述

1.1 概述

QT8Sxxx 系列指令集是一种精简指令集（RISC），指令宽度为 16 位，由操作码和 0~2 个操作数组成。指令按照功能可分为 5 类，即字节操作指令、位操作指令、立即数指令、分支指令、特殊控制指令。

每个指令周期由 4 个振荡周期组成，除非条件测试结果为真或指令执行改变了程序计数器的值，否则执行所有的指令都只需要一个指令周期。对于上述两种特征情况，指令执行需要两个指令周期。

任何一条指定文件寄存器作为指令一部分的指令都进行读-修改-写操作。读寄存器、修改数据并根据指令或目标标识符“d”存储结果。即使是写寄存器的指令也将先对改寄存器进行读操作。

1.2 符号说明

符号	范围	说明	符号	范围	说明
R/r	0-0x1ff	寄存器地址	C	-	进位标志
A	-	ACC 寄存器	DC	-	半进位标志
B/b	0-7	位地址	Z	-	零标志
I/i	0-0xff	立即数	d	0-1	目的操作数定义
K/k	0-0x1fff	标号	GIE	-	总中断使能位
TOS	-	栈顶	stkp	-	堆栈指针
PC	-	PC 指针			

2 QT8Sxxx 指令集表

指令集表中, d=1,目的操作数为 R; d=0,目的操作数为 A

指令类型	助记符	指令说明	周期数	影响标志位	备注
寄存器操作指令	ADDAR R,d	$R+A \rightarrow d$	1	Z,DC,C	
	ADCAR R,d	$R+A+C \rightarrow d$	1	Z,DC,C	
	SUBAR R,d	$A-R \rightarrow d$	1	Z,DC,C	
	SBCAR R,d	$A-R-C \rightarrow d$	1	Z,DC,C	
	SUBRA R,d	$R-A \rightarrow d$	1	Z,DC,C	
	SBCRA R,d	$R-A-C \rightarrow d$	1	Z,DC,C	
	ANDAR R,d	$R\&A \rightarrow d$	1	Z	
	ORAR R,d	$R A \rightarrow d$	1	Z	
	XORAR R,d	$R\wedge A \rightarrow d$	1	Z	
	COMR R,d	$R \rightarrow d$	1	Z	
	MOVR R,d	$R \rightarrow d$	1	Z	
	MOVAR R	$A \rightarrow R$	1	-	
	CLRR R	$0 \rightarrow R$	1	Z	
	SWAPR R,d	R 半字节交换 $\rightarrow d$	1	-	
	RLLR R,d	$R[7] \rightarrow C, \{R[6:0], C\} \rightarrow d$	1	C	
	RLLRNC R,d	$\{R[6:0], 0\} \rightarrow d$	1	-	
	RRR R,d	$R[0] \rightarrow C, \{C, R[7:1]\} \rightarrow d$	1	C	
	RRRNC R,d	$\{0, R[7:1]\} \rightarrow d$	1	-	
	DECR R,d	$R-1 \rightarrow d$	1	Z	
	DJZR R,d	$R-1 \rightarrow d, \text{SKIP if } 0$	1(2)	-	
	INCR R,d	$R+1 \rightarrow d$	1	Z	
	JZR R,d	$R+1 \rightarrow d, \text{SKIP if } 0$	1(2)	-	
	JNZR R,d	$R+1 \rightarrow d, \text{SKIP if } !0$	1(2)	-	(*1)
	DJNZR R,d	$R-1 \rightarrow d, \text{SKIP if } !0$	1(2)	-	(*1)
	JCMPPAR R	SKIP if $A=R$	1(2)	Z,C	(*1)
	JNCMPAR R	SKIP if $A \neq R$	1(2)	Z,C	(*1)
JGAR R	SKIP if $A \geq R$	1(2)	Z,C	(*1)	
JLAR R	SKIP if $A < R$	1(2)	Z,C	(*1)	
XCHAR R	$A \leftrightarrow R$	1	-	(*1)	
位操作指令	JBTS0 R,b	SKIP if $R[b]=0$	1(2)	-	
	JBTS1 R,b	SKIP if $R[b]=1$	1(2)	-	
	BCLR R,b	$0 \rightarrow R[b]$	1	-	
	BSET R,b	$1 \rightarrow R[b]$	1	-	

指令类型	助记符	指令说明	周期数	影响标志位	备注
立即数操作指令	ADDIA I	I+A → A	1	Z,DC,C	
	ADCIA I	I+A+C → A	1	Z,DC,C	(*2)
	SUBIA I	I-A → A	1	Z,DC,C	
	SBCIA I	I-A-C → A	1	Z,DC,C	
	SUBAI I	A-I → A	1	Z,DC,C	
	SBCAI I	A-I-C → A	1	Z,DC,C	
	ANDIA I	A&I → A	1	Z	
	ORIA I	A I → A	1	Z	
	XORIA I	A^I → A	1	Z	
	MOVIA I	I → A	1	-	
	RETIA I	Stack → PC, I → A	2	-	
	JCMPAI I	SKIP if A=I	1(2)	Z,C	(*1)
	JNCPAI I	SKIP if A≠I	1(2)	Z,C	(*1)
	特殊操作指令	RLA	A[7] → C, {A[6:0],C} → A	1	C
RLANC		{A[6:0],0} → A	1	-	
RRA		A[0] → C, {C,A[7:1]} → A	1	C	
RRANC		{0,A[7:1]} → A	1	-	
DECA		A-1 → A	1	Z	
DJZA		A-1 → A, SKIP if 0	1(2)	-	
INCA		A+1 → A	1	-	
JZA		A+1 → A, SKIP if 0	1(2)	-	
RETIE		Stack → PC, 1 → GIE	2	-	
RETURN		Stack → PC	2	-	
NOP		None Operation	1	-	
RDT		ROM[{{fsr1,fsr0}}] → {HBUF, A}	3	-	
DAA		加法后十进制调整	1	DC, C	
DSA		减法后十进制调整	1	DC, C	
PUSH		A, STATUS 压栈	1	-	(*1)(*2)
POP		A, STATUS 出栈	1	Z, DC, C	(*1)(*2)
CLRWDT		清除 WDT 寄存器	1	PD, TO	(*1)
分支指令		CALL I	I → PC, PC → Stack	2	-
	GOTO I	I → PC	2	-	
	ACALL I	I → PC, PC → Stack	2	-	
	AGOTO I	I → PC	2	-	

3 QT8Sxxx 指令说明

使用汇编编程时请先参考《编译器 语法解析》文件；可到官网下载此文件。

- A-----ACC 寄存器。
- R-----寄存器地址。
- Z-----状态寄存器 STATUS 的零标志；
- DC-----状态寄存器 STATUS 的辅助进位标志；
- C-----状态寄存器 STATUS 的进位标志。

ADDAR R,d

说明： R 与 A 相加
 操作数： $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作： $(R) + (A) \rightarrow$ (目标寄存器)
 受影响的状态位： Z, DC, C
 描述： 将寄存器 R 的数据与寄存器 A 的数据相加；
 如果 d 为 0, 结果存入寄存器 A；
 如果 d 为 1, 结果存入寄存器 R。
 周期： 1
 举例：
 REG_R EQU 0x00 ;定义寄存器 REG_R 的地址
 0x00;MOVIA 0x66 ;将数据 0x66 送入寄存器 A
 里面；
 MOVAR REG_R ;将寄存器 A 里面的数据送入寄存器 REG_R 里面。
 MOVIA 0x01
 ADDAR REG_R,0 ;将寄存器 REG_R 的数据与寄存器 A 的数据相加，结果存入 A 里面。
 MOVIA 0x01
 ADDAR REG_R,1 ;将寄存器 REG_R 的数据与寄存器 A 的数据相加，结果存入 REG_R 里
 面。注：上例中 ADDAR R,d 中 R 代表寄存器 REG_R，A 代表寄存器 ACC，后续指令中 R、A 也是同
 样的意思；

后续指令就不再做说明和举例。

例	指令 寄存器及标志位	ADDAR R,1					ADDAR R,0				
		R	A	Z	DC	C	R	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	0	0	0x00	0x00	1	0	0
例 2	指令执行前	0x08	0x08	0	0	0	0x08	0x08	0	0	0
	指令执行后	0x10	0x08	0	1	0	0x08	0x10	0	1	0
例 3	指令执行前	0x90	0x90	0	0	0	0x90	0x90	0	0	0
	指令执行后	0x20	0x90	0	0	1	0x90	0x20	0	0	1

ADCAR R,d

说明: R 与 A 带进位相加
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R) + (A) + (C) \rightarrow$ (目标寄存器)
 受影响的状态位: Z, DC, C
 说明: 将寄存器 R 的内容与寄存器 A 及进位值相加;
 如果 d 为 0, 结果存入寄存器 A;
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	ADCAR R,1					ADCAR R,0				
	寄存器及标志位	R	A	Z	DC	C	R	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	0	0	0x00	0x00	1	0	0
例 2	指令执行前	0x08	0x08	0	0	0	0x08	0x08	0	0	0
	指令执行后	0x10	0x08	0	1	0	0x08	0x10	0	1	0
例 3	指令执行前	0x90	0x90	0	0	0	0x90	0x90	0	0	0
	指令执行后	0x20	0x90	0	0	1	0x90	0x20	0	0	1

SUBAR R,d

说明: A 减去 R
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(A) - (R) \rightarrow$ (目标寄存器)
 受影响的状态位: Z, DC, C
 说明: 按二进制补码方式, 用寄存器 A 内容减去寄存器 R 的内容;
 如果 d 为 0, 结果存入寄存器 A;
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	SUBAR R,1					SUBAR R,0				
	寄存器及标志位	R	A	Z	DC	C	R	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	1	1	0x00	0x00	1	1	1
例 2	指令执行前	0x80	0x40	0	0	0	0x80	0x40	0	0	0
	指令执行后	0xC0	0x40	0	1	0	0x80	0xC0	0	1	0
例 3	指令执行前	0x09	0x10	0	0	0	0x09	0x10	0	0	0
	指令执行后	0x07	0x10	0	0	1	0x09	0x07	0	0	1

SBCAR R,d

说明: A 带借位减 R

操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(A) - (R) - (\bar{}) \rightarrow$ (目标寄存器)

受影响的状态位: Z, DC, C

说明: 按二进制补码方式,用寄存器 A 内容减去寄存器 R 的内容;
 如果 C = 0, 说明前次运算有借位, 则需再减去 1, 否则不需要再减。
 如果 d 为 0, 结果存入寄存器 A;
 如果 d 为 1, 结果存入寄存器 R.

周期: 1

举例:

例	指令 寄存器及标志位	SBCAR R,1					SBCAR R,0				
		R	A	Z	DC	C	R	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	1	0x00	0x00	0	0	1
	指令执行后	0x00	0x00	1	1	1	0x00	0x00	1	1	1
例 2	指令执行前	0x80	0x41	0	0	0	0x80	0x41	0	0	0
	指令执行后	0xC0	0x41	0	1	0	0x80	0xC0	0	1	0
例 3	指令执行前	0x09	0x10	0	0	0	0x09	0x10	0	0	0
	指令执行后	0x06	0x10	0	0	1	0x09	0x06	0	0	1

SBCRA R,d

说明: R 带借位减 A

操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(R) - (A) - (\bar{}) \rightarrow$ (目标寄存器)

受影响的状态位: Z, DC, C

说明: 按二进制补码方式,用寄存器 R 内容减去寄存器 A 的内容。
 如果 C = 0, 说明前次运算有借位, 则需再减去 1; 否则不需要再减。如
 果 d 为 0, 结果存入寄存器 A。
 如果 d 为 1, 结果存入寄存器 R.

周期: 1

举例:

例	指令 寄存器及标志位	SBCRA R,1					SBCRA R,0				
		R	A	Z	DC	C	R	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	1	1	0x00	0x00	1	1	1
例 2	指令执行前	0x41	0x80	0	0	0	0x41	0x80	0	0	0
	指令执行后	0xC0	0x80	0	1	0	0x41	0xC0	0	1	0
例 3	指令执行前	0x10	0x09	0	0	0	0x10	0x09	0	0	0
	指令执行后	0x06	0x09	0	0	1	0x10	0x06	0	0	1

SUBRA R,d

说明: R 减去 A
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R) - (A) \rightarrow$ (目标寄存器)
 受影响的状态位: Z, DC, C
 说明: 按二进制补码方式, 用寄存器 R 内容减去寄存器 A 的内容;
 如果 d 为 0, 结果存入寄存器 A;
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	SUBRA R,1					SUBRA R,0				
	寄存器及标志位	R	A	Z	DC	C	R	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	1	1	0x00	0x00	1	1	1
例 2	指令执行前	0x40	0x80	0	0	0	0x40	0x80	0	0	0
	指令执行后	0xC0	0x80	0	1	0	0x40	0xC0	0	1	0
例 3	指令执行前	0x10	0x09	0	0	0	0x10	0x09	0	0	0
	指令执行后	0x07	0x09	0	0	1	0x10	0x07	0	0	1

ANDAR R,d

说明: R 与 A 做逻辑与运算
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R) \text{ AND } (A) \rightarrow$ (目标寄存器)
 受影响的状态位: Z
 说明: 将寄存器 R 的内容与寄存器 A 进行逻辑与运算.
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	ANDAR R,1			ANDAR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0x01	0x00	0	0x01	0x00	0
	指令执行后	0x00	0x00	1	0x01	0x00	1
例 2	指令执行前	0x01	0x01	0	0x01	0x01	0
	指令执行后	0x01	0x01	0	0x01	0x01	0

ORAR R,d

说明: R 与 A 做逻辑或运算
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: (R) OR (A) → (目标寄存器)
 受影响的状态位: Z
 说明: 将寄存器 R 的内容与寄存器 A 进行逻辑或运算.
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	ORAR R,1			ORAR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0x00	0x00	0	0x00	0x00	0
	指令执行后	0x00	0x00	1	0x00	0x00	1
例 2	指令执行前	0x01	0x00	0	0x01	0x00	0
	指令执行后	0x01	0x00	0	0x01	0x01	0

XORAR R,d

说明: R 与 A 做逻辑异或运算
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: (R) XOR (A) → (目标寄存器)
 受影响的状态位: Z
 说明: 将寄存器 R 的内容与寄存器 A 进行逻辑异或运算.
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	XORAR R,1			XORAR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0x01	0x01	0	0x01	0x01	0
	指令执行后	0x00	0x01	1	0x01	0x00	1
例 2	指令执行前	0x01	0x00	0	0x01	0x00	0
	指令执行后	0x01	0x00	0	0x01	0x01	0

COMR R,d

说明: 对 R 取反
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(\bar{()}) \rightarrow$ (目标寄存器)
 受影响的状态位: Z
 说明: 将寄存器 R 的内容取反.
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.

周期: 1
 举例:

例	指令	COMR R,1			COMR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0xFF	0x01	0	0xFF	0x01	0
	指令执行后	0x00	0x01	1	0xFF	0x00	1
例 2	指令执行前	0x00	0x01	0	0x00	0x01	0
	指令执行后	0xFF	0x01	0	0x00	0xFF	0

MOVR R,d

说明: 传送 R
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(R) \rightarrow$ (目标寄存器)
 受影响的状态位: Z
 说明: 根据 d 的值将寄存器 R 的内容传送到目标寄存器.
 如果 d 为 0, 传送到寄存器 A.
 如果 d 为 1, 传送到寄存器 R.
 因为状态标志位 Z 会受影响, 所以可用 $d = 1$ 对寄存器 R 进行测试.

周期: 1
 举例:

例	指令	MOVR R,1			MOVR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0x00	0x01	0	0x00	0x01	0
	指令执行后	0x00	0x01	1	0x00	0x00	1
例 2	指令执行前	0x02	0x01	0	0x02	0x01	0
	指令执行后	0x02	0x01	0	0x02	0x02	0

MOVAR R

说明: 将 A 的内容传送到 R
 操作数: $0 \leq R \leq 0x1ff$
 操作: (A) → (R)
 受影响的状态位: 无
 说明: 将寄存器 A 的内容传送到寄存器 R.
 周期: 1
 举例:

例	指令	MOVAR R	
	寄存器及标志位	R	A
例 1	指令执行前	0x00	0x01
	指令执行后	0x01	0x01

CLRR R

说明: 将 R 清零
 操作数: $0 \leq R \leq 0x1ff$
 操作: $0x00 \rightarrow (R)$
 $1 \rightarrow Z$
 受影响的状态位: Z
 说明: 寄存器 R 的内容清零,Z 位置 1
 周期: 1
 举例:

例	指令	CLRR R	
	寄存器及标志位	R	Z
例 1	指令执行前	0xFF	0
	指令执行后	0x00	1

SWAPR R,d

说明：将 R 的高半字节和低半字节交换

操作数： $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作： $(R[3:0]) \rightarrow$ (目标寄存器[7:4])
 $(R[7:4]) \rightarrow$ (目标寄存器[3:0])

受影响的状态位：无

说明：寄存器 R 内容的高半字节和低半字节相互交换，
 如果 d 为 0，结果存入寄存器 A。
 如果 d 为 1，结果存入寄存器 R。

周期：1

举例：

例	指令	SWAPR R,1		SWAPR R,0	
	寄存器及标志位	R	A	R	A
例 1	指令执行前	0xF0	0x01	0xF0	0x01
	指令执行后	0x0F	0x01	0xF0	0x0F

RLR R,d

说明：R 带进位循环左移

操作数： $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作： $(R[7]) \rightarrow$ (C)
 $(\{R[6:0],(C)\}) \rightarrow$ (目标寄存器)

受影响的状态位：C

说明：将寄存器 R 的内容连同进位值一起循环左移 1 位
 如果 d 为 0，结果存入寄存器 A。
 如果 d 为 1，结果存入寄存器 R。

周期：1

举例：

例	指令	RLR R,1			RLR R,0		
	寄存器及标志位	R	A	C	R	A	C
例 1	指令执行前	0x80	0x01	0	0x80	0x01	0
	指令执行后	0x00	0x01	1	0x80	0x00	1
例 2	指令执行前	0x40	0x01	1	0x40	0x01	1
	指令执行后	0x81	0x01	0	0x40	0x81	0

RLRNC R,d

说明: R 不带进位左移
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(\{R[6:0],0\}) \rightarrow$ (目标寄存器)
 受影响的状态位: 无
 说明: 将寄存器 R 的内容左移 1 位, 最低位补 0
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	RLRNC R,1		RLRNC R,0	
	寄存器及标志位	R	A	R	A
例 1	指令执行前	0x40	0x01	0x40	0x01
	指令执行后	0x80	0x01	0x40	0x80

RRR R,d

说明: R 带进位循环右移
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R[0]) \rightarrow (C)$
 $(\{(C), R[7:1]\}) \rightarrow$ (目标寄存器)
 受影响的状态位: C
 说明: 将寄存器 R 的内容连同进位值一起循环右移 1 位
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	RRR R,1			RRR R,0		
	寄存器及标志位	R	A	C	R	A	C
例 1	指令执行前	0x01	0x02	0	0x01	0x02	0
	指令执行后	0x00	0x02	1	0x01	0x00	1
例 2	指令执行前	0x80	0x02	1	0x80	0x02	1
	指令执行后	0xC0	0x02	0	0x80	0xC0	0

RRRNC R,d

说明: R 不带进位右移
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(\{0, R[7:1]\}) \rightarrow$ (目标寄存器)
 受影响的状态位: 无
 说明: 将寄存器 R 的内容右移 1 位, 最高位补 0
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	RRRNC R,1		RRRNC R,0	
	寄存器及标志位	R	A	R	A
例 1	指令执行前	0x80	0x02	0x80	0x02
	指令执行后	0x40	0x02	0x80	0x40

DECR R,d

说明: R 递减 1
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R) - 1 \rightarrow$ (目标寄存器)
 受影响的状态位: Z
 说明: 将寄存器 R 的内容递减 1.
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	DECR R,1			DECR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0x01	0x03	0	0x01	0x03	0
	指令执行后	0x00	0x03	1	0x01	0x00	1
例 2	指令执行前	0x02	0x03	0	0x02	0x03	0
	指令执行后	0x01	0x03	0	0x02	0x01	0

JNZR R,d

说明: R 递增 1, 非 0 则跳过

操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(R) + 1 \rightarrow$ (目标寄存器)
 如果结果 $\neq 0$ 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, 结果存入寄存器 A; 如果 d 为 1, 结果存入寄存器 R.
 如果结果为 0, 则执行下一条指令; 如果结果不为 0, 则跳过执行下一条指令(即执行该指令下的第二条指令)。

周期: 1or2.

举例:

例	指令	JNZR R,1			JNZR R,0		
	寄存器及标志位	R	A	执行情况	R	A	执行情况
例 1	指令执行前	0xFF	0x03	执行下一	0xFF	0x03	执行下一
	指令执行后	0x00	0x03	条指令	0xFF	0x00	条指令
例 2	指令执行前	0x01	0x03	跳过下一	0x01	0x03	跳过下一
	指令执行后	0x02	0x03	条指令	0x01	0x02	条指令

DJZR R,d

说明: R 递减 1, 为 0 则跳过

操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(R) - 1 \rightarrow$ (目标寄存器)
 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, 结果存入寄存器 A; 如果 d 为 1, 结果存入寄存器 R.
 如果结果不为 0, 则执行下一条指令; 如果结果为 0, 则跳过执行下一条指令(即执行该指令下的第二条指令)。

周期: 1or2.

举例:

例	指令	DJZR R,1			DJZR R,0		
	寄存器及标志位	R	A	执行情况	R	A	执行情况
例 1	指令执行前	0x02	0x00	执行下一	0x02	0x00	执行下一
	指令执行后	0x01	0x00	条指令	0x02	0x01	条指令
例 2	指令执行前	0x01	0x00	跳过下一	0x01	0x00	跳过下一
	指令执行后	0x00	0x00	条指令	0x00	0x00	条指令

INCR R,d

说明: R 递增 1
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R) + 1 \rightarrow$ (目标寄存器)
 受影响的状态位: Z
 说明: 将寄存器 R 的内容递增 1.
 如果 d 为 0, 结果存入寄存器 A.
 如果 d 为 1, 结果存入寄存器 R.
 周期: 1
 举例:

例	指令	INCR R,1			INCR R,0		
	寄存器及标志位	R	A	Z	R	A	Z
例 1	指令执行前	0xFF	0x03	0	0xFF	0x03	0
	指令执行后	0x00	0x03	1	0xFF	0x00	1
例 2	指令执行前	0x01	0x03	0	0x01	0x03	0
	指令执行后	0x02	0x03	0	0x01	0x02	0

JZR R,d

说明: R 递增 1, 为 0 则跳过
 操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$
 操作: $(R) + 1 \rightarrow$ (目标寄存器)
 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, 结果存入寄存器 A; 如果 d 为 1, 结果存入寄存器 R.
 如果结果不为 0, 则执行下一条指令; 如果结果为 0, 则跳过执行下一条指令(即执行该指令下的第二条指令).
 周期: 1or2.
 举例:

例	指令	JZR R,1			JZR R,0		
	寄存器及标志位	R	A	执行情况	R	A	执行情况
例 1	指令执行前	0x01	0x03	执行下一	0x01	0x03	执行下一
	指令执行后	0x02	0x03	条指令	0x01	0x02	条指令
例 2	指令执行前	0xFF	0x03	跳过下一	0xFF	0x03	跳过下一
	指令执行后	0x00	0x03	条指令	0xFF	0x00	条指令

DJNZR R,d

说明: R 递减 1, 非 0 则跳过

操作数: $0 \leq R \leq 0x1ff$
 $d \in [0,1]$

操作: $(R) - 1 \rightarrow$ (目标寄存器)
 如果结果 $\neq 0$ 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, 结果存入寄存器 A; 如果 d 为 1, 结果存入寄存器 R.
 如果结果为 0, 则执行下一条指令; 如果结果不为 0, 则跳过执行下一条指令(即执行该指令下的第二条指令)。

周期: 1or2.

举例:

例	指令	DJNZR R,1			DJNZR R,0		
	寄存器及标志位	R	A	执行情况	R	A	执行情况
例 1	指令执行前	0x01	0x03	执行下一	0x01	0x03	执行下一
	指令执行后	0x00	0x03	条指令	0x01	0x00	条指令
例 2	指令执行前	0x02	0x03	跳过下一	0x02	0x03	跳过下一
	指令执行后	0x01	0x03	条指令	0x02	0x01	条指令

JCMPPAR R

说明: 比较 A 与 R, 相等则跳过

操作数: $0 \leq R \leq 0x1ff$

操作: $(A) \text{ Vs } (R)$
 如果 $A = R$ 则跳过

受影响的状态位: Z, C

说明: 将寄存器 A 的内容与寄存器 R 比较.
 如果 $A \neq R$, 则执行下一条指令; 如果 $A = R$, 则跳过执行下一条指令(即执行该指令下的第二条指令)。

周期: 1or2.

举例:

例	指令	JCMPPAR R				执行情况
	寄存器及标志位	R	A	Z	C	
例 1	指令执行前	0x01	0x02	0	0	执行下一
	指令执行后	0x01	0x02	0	1	条指令
例 2	指令执行前	0x02	0x01	0	0	执行下一
	指令执行后	0x02	0x01	0	0	条指令
例 3	指令执行前	0x01	0x01	0	0	跳过下一
	指令执行后	0x01	0x01	1	1	条指令

JNCMPAR R

说明: 比较 A 与 R, 不等则跳过
 操作数: $0 \leq R \leq 0x1ff$
 操作: (A) Vs (R)
 如果 $A \neq R$ 则跳过
 受影响的状态位: Z, C
 说明: 将寄存器 A 的内容与寄存器 R 比较.
 如果 $A = R$, 则执行下一条指令; 如果 $A \neq R$, 则跳过执行下一条指令 (即执行该指令下的第二条指令).
 周期: 1or2.
 举例:

例	指令	JNCMPAR R				执行情况
	寄存器及标志位	R	A	Z	C	
例 1	指令执行前	0x01	0x01	0	0	执行下一条指令
	指令执行后	0x01	0x01	1	1	
例 2	指令执行前	0x02	0x01	0	0	执行下一条指令
	指令执行后	0x02	0x01	0	0	
例 3	指令执行前	0x01	0x02	0	0	跳过下一条指令
	指令执行后	0x01	0x02	0	1	

JGAR R

说明: 比较 A 与 R, A 大于等于 R 则跳过
 操作数: $0 \leq R \leq 0x1ff$
 操作: (A) Vs (R)
 如果 $A \geq R$ 则跳过
 受影响的状态位: Z, C
 说明: 将寄存器 A 的内容与寄存器 R 比较.
 如果 $A < R$, 则执行下一条指令; 如果 $A \geq R$, 则跳过执行下一条指令 (即执行该指令下的第二条指令).
 周期: 1or2.
 举例:

例	指令	JGAR R				执行情况
	寄存器及标志位	R	A	Z	C	
例 1	指令执行前	0x02	0x01	0	0	执行下一条指令
	指令执行后	0x02	0x01	0	0	
例 2	指令执行前	0x01	0x02	0	0	跳过下一条指令
	指令执行后	0x01	0x02	0	1	
例 3	指令执行前	0x01	0x01	0	0	跳过下一条指令
	指令执行后	0x01	0x01	1	1	

JLAR R

说明: 比较 A 与 R, A 小于 R 则跳过

操作数: $0 \leq R \leq 0x1ff$

操作: (A) Vs (R)
如果 $A < R$ 则跳过

受影响的状态位: Z, C

说明: 将寄存器 A 的内容与寄存器 R 比较.
如果 $A \geq R$, 则执行下一条指令; 如果 $A < R$, 则跳过执行下一条指令(即执行该指令下的第二条指令)。

周期: 1or2.

举例:

例	指令	JLAR R				执行情况
	寄存器及标志位	R	A	Z	C	
例 1	指令执行前	0x01	0x02	0	0	执行下一条指令
	指令执行后	0x01	0x02	0	1	
例 2	指令执行前	0x01	0x01	0	0	执行下一条指令
	指令执行后	0x01	0x01	1	1	
例 3	指令执行前	0x02	0x01	0	0	跳过下一条指令
	指令执行后	0x02	0x01	0	0	

XCHAR R

说明: A 与 R 交换

操作数: $0 \leq R \leq 0x1ff$

操作: (R) → (A)
(A) → (R)

受影响的状态位: 无

说明: 将寄存器 A 的内容与寄存器 R 交换.

周期: 1

举例:

例	指令	XCHAR R	
	寄存器及标志位	R	A
例 1	指令执行前	0x01	0x02
	指令执行后	0x02	0x01

JBTS0 R,b

说明: 测试 R 的指定位, 为 0 则跳过

操作数: $0 \leq R \leq 0x1ff$
 $0 \leq b \leq 0x7$

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 $b = 1$, 则执行下一条指令.
 如果寄存器 R 的位 $b = 0$, 则跳过执行下一条指令(即执行该指令下的第二条指令).

周期: 1or2.

举例:

例	指令	JBTS0 R,0		JBTS0 R,7	
	寄存器及标志位	R	执行情况	R	执行情况
例 1	指令执行前	00000001	执行下一	10000000	执行下一
	指令执行后	00000001	条指令	10000000	条指令
例 2	指令执行前	00000010	跳过下一	01000000	跳过下一
	指令执行后	00000010	条指令	01000000	条指令

JBTS1 R,b

说明: 测试 R 的指定位, 为 1 则跳过

操作数: $0 \leq R \leq 0x1ff$
 $0 \leq b \leq 0x7$

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 $b = 1$, 则跳过执行下一条指令(即执行该指令下的第二条指令).
 如果寄存器 R 的位 $b = 0$, 则执行下一条指令.

周期: 1or2.

举例:

例	指令	JBTS1 R,0		JBTS1 R,7	
	寄存器及标志位	R	执行情况	R	执行情况
例 1	指令执行前	00000010	执行下一	01000000	执行下一
	指令执行后	00000010	条指令	01000000	条指令
例 2	指令执行前	00000001	执行下一	10000000	执行下一
	指令执行后	00000001	条指令	10000000	条指令

BCLR R,b

说明: 将 R 的指定位清零
 操作数: $0 \leq R \leq 0x1ff$
 $0 \leq b \leq 0x7$
 操作: $0 \rightarrow (R[b])$
 受影响的状态位: 无
 说明: 将寄存器 R 的位 b 清零
 周期: 1
 举例:

例	指令	BCLR R,0	BCLR R,7
	寄存器及标志位	R	R
例 1	指令执行前	00000001	10000000
	指令执行后	00000000	00000000

BSET R,b

说明: 将 R 的指定位置 1
 操作数: $0 \leq R \leq 0x1ff$
 $0 \leq b \leq 0x7$
 操作: $1 \rightarrow (R[b])$
 受影响的状态位: 无
 说明: 将寄存器 R 的位 b 置 1
 周期: 1
 举例:

例	指令	BSET R,0	BSET R,7
	寄存器及标志位	R	R
例 1	指令执行前	00000000	00000000
	指令执行后	00000001	10000000

ADDIA I

说明: I 与 A 相加
 操作数: $0 \leq I \leq 0xff$
 操作: $(I) + (A) \rightarrow (A)$
 受影响的状态位: Z, DC, C
 说明: 将立即数 I 与寄存器 A 相加. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	ADDIA I				
	寄存器及标志位	I	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	0	0
例 2	指令执行前	0x08	0x08	0	0	0
	指令执行后	0x08	0x10	0	1	0
例 3	指令执行前	0x90	0x90	0	0	0
	指令执行后	0x90	0x20	0	0	1

ADCIA I

说明: I 与 A 带进位相加
 操作数: $0 \leq I \leq 0xff$
 操作: $(I) + (A) + (C) \rightarrow (A)$
 受影响的状态位: Z, DC, C
 说明: 将立即数 I 与寄存器 A 相加. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	ADCIA I				
	寄存器及标志位	I	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	0	0
例 2	指令执行前	0x08	0x08	0	0	0
	指令执行后	0x08	0x10	0	1	0
例 3	指令执行前	0x90	0x90	0	0	0
	指令执行后	0x90	0x20	0	0	1

SUBIA I

说明: I 减去 A
 操作数: $0 \leq I \leq 0xff$
 操作: $(I) - (A) \rightarrow (A)$
 受影响的状态位: Z, DC, C
 说明: 按二进制补码方式, 用立即数 I 减去寄存器 A, 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	SUBIA I				
	寄存器及标志位	I	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	1	1
例 2	指令执行前	0x40	0x80	0	0	0
	指令执行后	0x40	0xC0	0	1	0
例 3	指令执行前	0x10	0x09	0	0	0
	指令执行后	0x10	0x07	0	0	1

SBCIA I

说明: I 带借位减 A
 操作数: $0 \leq I \leq 0xff$
 操作: $(I) - (A) - (\bar{C}) \rightarrow (A)$
 受影响的状态位: Z, DC, C
 说明: 按二进制补码方式, 用立即数 I 减去寄存器 A 的内容. 如果 $C = 0$, 说明前次运算有借位, 则需再减去 1; 否则不需要再减. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	SBCIA I				
	寄存器及标志位	I	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	1
	指令执行后	0x00	0x00	1	1	1
例 2	指令执行前	0x41	0x80	0	0	0
	指令执行后	0x41	0xC0	0	1	0
例 3	指令执行前	0x10	0x09	0	0	0
	指令执行后	0x10	0x06	0	0	1

SUBAI I

说明: A 减去 I
 操作数: $0 \leq I \leq 0xff$
 操作: $(A) - (I) \rightarrow (A)$
 受影响的状态位: Z, DC, C
 说明: 按二进制补码方式, 用寄存器 A 内容减去立即数 I, 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	SUBAI I				
	寄存器及标志位	I	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	0
	指令执行后	0x00	0x00	1	1	1
例 2	指令执行前	0x80	0x40	0	0	0
	指令执行后	0x80	0xC0	0	1	0
例 3	指令执行前	0x09	0x10	0	0	0
	指令执行后	0x09	0x07	0	0	1

SBCAI I

说明: A 带借位减 I
 操作数: $0 \leq I \leq 0xff$
 操作: $(A) - (I) - (\bar{C}) \rightarrow (A)$
 受影响的状态位: Z, DC, C
 说明: 按二进制补码方式, 用 A 寄存器内容减去立即数 I. 如果 C = 0, 说明前次运算有借位, 则需再减去 1; 否则不需要再减.
 结果存入 A 寄存器.
 周期: 1
 举例:

例	指令	SBCAI I				
	寄存器及标志位	I	A	Z	DC	C
例 1	指令执行前	0x00	0x00	0	0	1
	指令执行后	0x00	0x00	1	1	1
例 2	指令执行前	0x80	0x41	0	0	0
	指令执行后	0x80	0xC0	0	1	0
例 3	指令执行前	0x09	0x10	0	0	0
	指令执行后	0x09	0x06	0	0	1

ANDIA I

说明: A 与 I 做逻辑与运算
 操作数: $0 \leq I \leq 0xff$
 操作: (A) AND (I) \rightarrow (A)
 受影响的状态位: Z
 说明: 将寄存器 A 的内容与立即数 I 进行逻辑与运算. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	ANDIA I		
	寄存器及标志位	I	A	Z
例 1	指令执行前	0x00	0x01	0
	指令执行后	0x00	0x00	1
例 2	指令执行前	0x01	0x01	0
	指令执行后	0x01	0x01	0

ORIA I

说明: A 与 I 做逻辑或运算
 操作数: $0 \leq I \leq 0xff$
 操作: (A) OR (I) \rightarrow (A)
 受影响的状态位: Z
 说明: 将寄存器 A 的内容与立即数 I 进行逻辑或运算. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	ORIA I		
	寄存器及标志位	I	A	Z
例 1	指令执行前	0x00	0x00	0
	指令执行后	0x00	0x00	1
例 2	指令执行前	0x01	0x00	0
	指令执行后	0x01	0x01	0

XORIA I

说明: A 与 I 做逻辑异或运算
 操作数: $0 \leq I \leq 0xff$
 操作: (A) XOR (I) \rightarrow (A)
 受影响的状态位: Z
 说明: 将寄存器 A 的内容与立即数 I 进行逻辑异或运算, 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	XORIA I		
	寄存器及标志位	I	A	Z
例 1	指令执行前	0x01	0x01	0
	指令执行后	0x01	0x00	1
例 2	指令执行前	0x01	0x00	0
	指令执行后	0x01	0x01	0

MOVIA I

说明: 将 I 传送到 A
 操作数: $0 \leq I \leq 0xff$
 操作: (I) \rightarrow (A)
 受影响的状态位: 无
 说明: 将立即数 I 传送到寄存器 A.
 周期: 1
 举例:

例	指令	MOVIA I	
	寄存器及标志位	I	A
例 1	指令执行前	0x01	0x00
	指令执行后	0x01	0x01

RETIA I

说明: 返回并将立即数 I 传送到 A
 操作数: $0 \leq I \leq 0xff$
 操作: (I) → (A)
 Stack → PC
 受影响的状态位: 无
 说明: 将立即数 I 传送到寄存器 A. 将栈顶的返回地址值传送到程序计数器.
 周期: 2
 举例:

例	指令	RETIA I	
	寄存器及标志位	I	A
例 1	指令执行前	0x01	0x00
	指令执行后	0x01	0x01

JCMPAI I

说明: 比较 A 与 I, 相等则跳过
 操作数: $0 \leq I \leq 0xff$
 操作: (A) Vs (I)
 如果 A = I 则跳过
 受影响的状态位: Z, C
 说明: 将寄存器 A 的内容与立即数 I 比较.
 如果 A ≠ I, 则执行下一条指令; 如果 A = I, 则跳过执行下一条指令(即执行该指令 下
 的第二条指令).
 周期: 1or2.
 举例:

例	指令	JCMPAI I				执行情况
	寄存器及标志位	I	A	Z	C	
例 1	指令执行前	0x02	0x01	0	0	执行下一 条指令
	指令执行后	0x02	0x01	0	0	
例 2	指令执行前	0x01	0x02	0	0	执行下一 条指令
	指令执行后	0x01	0x02	0	1	
例 3	指令执行前	0x01	0x01	0	0	跳过下一 条指令
	指令执行后	0x01	0x01	1	1	

JNCMPAI I

说明: 比较 A 与 I, 不等则跳过

操作数: $0 \leq I \leq 0xff$

操作: (A) Vs (I)
如果 $A \neq I$ 则跳过

受影响的状态位: Z, C

说明: 将寄存器 A 的内容与立即数 I 比较.
如果 $A = I$, 则执行下一条指令; 如果 $A \neq I$, 则跳过执行下一条指令(即执行该指令下的第二条指令)。

周期: 1or2.

举例:

例	指令	JNCMPAI I				执行情况
	寄存器及标志位	I	A	Z	C	
例 1	指令执行前	0x01	0x01	0	0	执行下一条指令
	指令执行后	0x01	0x01	1	1	
例 2	指令执行前	0x02	0x01	0	0	跳过下一条指令
	指令执行后	0x02	0x01	0	0	
例 3	指令执行前	0x01	0x02	0	0	跳过下一条指令
	指令执行后	0x01	0x02	0	1	

RLA

说明: A 带进位循环左移

操作数: 无

操作: (A[7]) → (C)
({A[6:0],(C)}) → (A)

受影响的状态位: C

说明: 将寄存器 A 的内容连同进位值一起循环左移 1 位, 结果存入寄存器 A.

周期: 1

举例:

例	指令	RLA	
	寄存器及标志位	A	C
例 1	指令执行前	0x80	0
	指令执行后	0x00	1
例 2	指令执行前	0x40	1
	指令执行后	0x81	0

RLANC

说明: A 不带进位左移
 操作数: 无
 操作: $(\{A[6:0],0\}) \rightarrow (A)$
 受影响的状态位: 无
 说明: 将寄存器 A 的内容左移 1 位, 最低位补 0. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	RLANC
	寄存器及标志位	A
例 1	指令执行前	0x80
	指令执行后	0x00

RRA

说明: A 带进位循环右移
 操作数: 无
 操作: $(A[0]) \rightarrow (C)$
 $(\{(C), A[7:1]\}) \rightarrow (A)$
 受影响的状态位: C
 说明: 将寄存器 A 的内容连同进位值一起循环右移 1 位. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	RRA	
	寄存器及标志位	A	C
例 1	指令执行前	0x01	0
	指令执行后	0x00	1
例 2	指令执行前	0x80	1
	指令执行后	0xC0	0

RRANC

说明: A 不带进位右移
 操作数: 无
 操作: $(\{0, A[7:1]\}) \rightarrow (A)$
 受影响的状态位: 无
 说明: 将寄存器 A 的内容右移 1 位, 最高位补 0. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	RRANC
	寄存器及标志位	A
例 1	指令执行前	0x01
	指令执行后	0x00

DECA

说明: A 递减 1
 操作数: 无
 操作: $(A) - 1 \rightarrow (A)$
 受影响的状态位: Z
 说明: 将寄存器 A 的内容递减 1. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	DECA	
	寄存器及标志位	A	Z
例 1	指令执行前	0x01	0
	指令执行后	0x00	1
例 2	指令执行前	0x02	0
	指令执行后	0x01	0

DJZA

说明: A 递减 1, 为 0 则跳过
 操作数: 无
 操作: $(A) - 1 \rightarrow (A)$
 如果结果=0 则跳过
 受影响的状态位: 无
 说明: 将寄存器 A 的内容递减 1;结果存入 A 寄存器.
 如果结果不为 0, 则执行下一条指令.
 如果结果为 0, 则跳过执行下一条指令(即执行该指令下的第二条指令).
 周期: 1or2.
 举例:

例	指令	DJZA	
	寄存器及标志位	A	执行情况
例 1	指令执行前	0x02	执行下一 条指令
	指令执行后	0x01	
例 2	指令执行前	0x01	跳过下一 条指令
	指令执行后	0x00	

INCA

说明: A 递增 1
 操作数: 无
 操作: $(A) + 1 \rightarrow (A)$
 受影响的状态位: 无
 说明: 将寄存器 A 的内容递增 1. 结果存入寄存器 A.
 周期: 1
 举例:

例	指令	INCA
	寄存器及标志位	A
例 1	指令执行前	0x01
	指令执行后	0x02

JZA

说明: A 递增 1, 为 0 则跳过
 操作数: 无
 操作: $(A) + 1 \rightarrow (A)$
 如果结果=0 则跳过
 受影响的状态位: 无
 说明: 将寄存器 A 的内容递增 1.;结果存入寄存器 A.
 如果结果不为 0, 则执行下一条指令.
 如果结果为 0, 则跳过执行下一条指令(即执行该指令下的第二条指令).
 周期: 1or2.
 举例:

例	指令	RDT	
	寄存器及标志位	A	执行情况
例 1	指令执行前	0x01	执行下一 条指令
	指令执行后	0x02	
例 2	指令执行前	0xFF	跳过下一 条指令
	指令执行后	0x00	

RETIE

说明: 从中断返回
 操作数: 无
 操作: $1 \rightarrow GIE$
 $Stack \rightarrow PC$
 受影响的状态位: 无
 说明: 全局中断允许位 GIE 置 1. 将栈顶的返回地址值传送到程序计数器.
 周期: 2
 举例:

RETURN

说明: 从子程序返回
操作数: 无
操作: Stack → PC
受影响的状态位: 无
说明: 从子程序返回, 执行出栈动作, 将栈顶的返回地址值传送到程序计数器.
周期: 2
举例:

NOP

说明: 空操作
操作数: 无
操作: 空操作
受影响的状态位: 无
说明: 不执行任何操作.
周期: 1
举例:

RDT

说明: 读表
 操作数: 无
 操作: ROM[{fsr1,fsr0}]高 8 位 → HBUF
 ROM[{fsr1,fsr0}]低 8 位 → A
 受影响的状态位: 无
 说明: 进行读表操作, 将{fsr1,fsr0}所指向的 ROM Code 高 8 位传送到寄存器 HBUF,低 8 位传送到 A 寄存器。
 周期: 3
 举例:

START:

```

        CLRR      TABDL
        CLRR      TABDH
        MOVIA     HIGH(DTAB) ;获取数据表地址高位
        MOVAR     FSR1      ;设置数据表高位指针
        MOVIA     LOW (DTAB) ;获取数据表地址地位
        MOVAR     FSR0      ;设置数据表低位指针
        RDT
        MOVAR     TABDL     ;将低位数据放在
        TABDLMOVR HBUF,A ;高位数据读入
        累加器 AMOVAR     TABDH
        RETURN
    
```

DTAB:

```

        DW      0x55AA
        DW      0xFFFF
    
```

例	指令	RDT	
	寄存器及标志位	TABDL	TABDH
例 1	指令执行前	0x00	0x00
	指令执行后	0xAA	0x55

DAA

说明: 加法后十进制调整
 操作数: 无
 操作: (A)加法十进制调整 → (A)
 受影响的状态位: DC, C
 说明: 将寄存器 A 的内容进行加法十进制调整, 结果存入寄存器 A(寄存器 A 的内容必须是十进制数, 否则会出错).
 周期: 1
 举例:

例	指令	ADDIA I			
	寄存器及标志位	I	A	DC	C
例 1	指令执行前	0x05	0x05	0	0
	指令执行后	0x00	0x10	1	0
例 2	指令执行前	0x60	0x60	0	0
	指令执行后	0x00	0x20	0	1

DSA

说明: 减法后十进制调整
 操作数: 无
 操作: (A)减法十进制调整 → (A)
 受影响的状态位: DC, C
 说明: 将寄存器 A 的内容进行减法十进制调整, 结果存入寄存器 A(寄存器 A 的内容必须是十进制数, 否则会出错).
 周期: 1
 举例:

例	指令	SUBIA I			
	寄存器及标志位	I	A	DC	C
例 1	指令执行前	0x05	0x10	0	0
	指令执行后	0x00	0x95	1	0
例 2	指令执行前	0x10	0x05	0	0
	指令执行后	0x00	0x05	0	1

PUSH

说明: 进栈
操作数: 无
操作: PC → Stack
保存 A, STATUS
受影响的状态位: 无
说明: 进栈指令, 程序计数器值存入栈顶, 保存寄存器 A 和 STATUS 寄存器值.
周期: 1

POP

说明: 出栈
操作数: 无
操作: Stack → PC
恢复 A, STATUS
受影响的状态位: Z, DC, C
说明: 出栈指令, 将栈顶的返回地址值传送到程序计数器, 恢复原保存的寄存器 A 和 STATUS 寄存器值.
周期: 1
举例:

CLRWDT

说明: 清零看门狗计数器
操作数: 无
操作: 0x00 → WDT
1 → TO
1 → PD
受影响的状态位: TO, PD
说明: 将看门狗计数器清零, 状态位 TO 和 PD 置 1.
周期: 1
举例:

CALL I

说明: 调用子程序
操作数: $0 \leq I \leq 0x3fff$
操作: PC → Stack
I → PC
受影响的状态位: 无
说明: 调用子程序, 将程序计数器存入栈顶, 将立即数 I 作为地址值传送到程序计数器
周期: 2
举例:

GOTO I

说明: 无条件跳转
操作数: $0 \leq I \leq 0x3fff$
操作: $I \rightarrow PC$
受影响的状态位: 无
说明: 无条件跳转, 将立即数 I 作为地址值传送到程序计数器
周期: 2
举例:

ACALL I

说明: 常调用子程序
操作数: $0 \leq I \leq 0x7fff$
操作: $PC \rightarrow Stack$
 $I \rightarrow PC$
受影响的状态位: 无
说明: 常调用子程序, 将程序计数器存入栈顶, 将立即数 I 作为地址值传送到程序计数器
周期: 2
举例:

AGOTO I

说明: 无条件常跳转
操作数: $0 \leq I \leq 0x7fff$
操作: $I \rightarrow PC$
受影响的状态位: 无
说明: 无条件常跳转, 将立即数 I 作为地址值传送到程序计数器
周期: 2
举例: